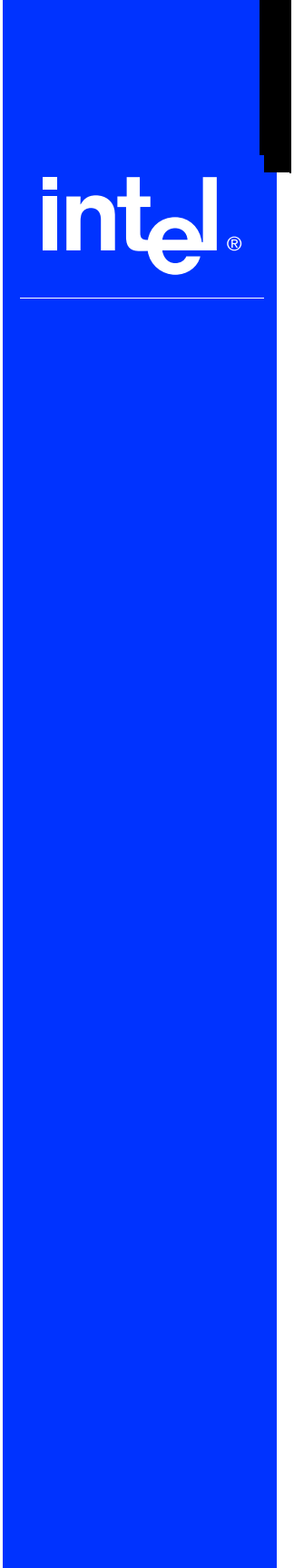

Intel® Integrated Performance Primitives for Intel® Architecture

Quick Reference

Part 1:
Signal Processing

Document Number: 254146-007US

World Wide Web: http://developer.intel.com

Version	Version Information	Date
-001	Original issue. Documents API included into Intel IPP release 4.0 . Links to Intel IPP Reference Manual (document number A24968-4002).	10/2003
-002	Documents API included into Intel IPP release 4.1 beta. Links to Intel IPP Reference Manual (document number A24968-012).	04/2004
-003	Documents API included into Intel IPP release 4.1. Links to Intel IPP Reference Manual (document number A24968-013).	07/2004
-004	Documents API included into Intel IPP release 5.0 beta. Links to Intel IPP Reference Manual (document number A24968-014).	03/2005
-005	Documents API included into Intel IPP release 5.0. Links to Intel IPP Reference Manual (document number A24968-015).	08/2005
-006	Documents API included into Intel IPP release 5.1 beta. Links to Intel IPP Reference Manual (document number A24968-016).	10/2005
-007	Documents API included into Intel IPP release 5.1. Links to Intel IPP Reference Manual (document number A24968-017US).	02/2006

Overview

This Quick Reference provides a full list of prototypes for all functions included into the signal processing domain of the Intel® Integrated Performance Primitives (Intel® IPP) for Intel® architecture.

The Intel IPP software is a new generation of the Intel® Performance Libraries, comprising a broad range of functions for basic software functionality and including, among many others, the signal processing functions domain.

Intel IPP is a cross architecture software library optimized for the latest generations of Intel microprocessors, including Intel® Pentium® 4 processor, Intel® Itanium® 2 processor, and Intel® PCA application processors.

The signal processing subset of Intel IPP includes the following functional domains:

Fixed-Accuracy Arithmetic Functions

Data Compression Functions

This Quick Reference contains an alphabetic list of all signal processing function names with hyperlinks to the full set of function prototypes available in the library and a short description of function operation. The detailed information about all functions, including introductory material, general argument description, function behavior explanation, return values, examples and more, is given in the Intel® Integrated Performance Primitives Reference Manual, volume 1 (document number A24968-017US).

Each function name or prototype, included into the reference section of this Quick Reference has a hyperlink to the respective section of the above Reference Manual.

For these links to work properly, please make sure that:

1. You have the correct version of the Intel IPP Reference Manual available with the Intel IPP distribution (file name ippsman.pdf), and
2. Files for the Quick Reference and for the Reference Manual reside in the same directory.

If you have jumped to the manual and wish to return back to the Quick Reference, use the Go to the Previous View button in the Adobe Acrobat* tool.

For more information about the Intel IPP library, please refer to the Web site at developer.intel.com/software/products/ipp/.

Alphabetic List of Routines

D

E

F

I

J

L

M

N

O

P

T

U

V

W

X

Z

Support Functions

Version Information Functions

ippsGetLibVersion

Returns information about the active version of Intel IPP signal processing software.

```
const IppLibraryVersion* ippsGetLibVersion(void);
```

Memory Allocation Functions

ippsMalloc

Allocates memory aligned to 32-byte boundary.

```
Ipp8u* ippsMalloc_8u(int len);
Ipp16u* ippsMalloc_16u(int len);
Ipp32u* ippsMalloc_32u(int len);
Ipp8s* ippsMalloc_8s(int len);
Ipp16s* ippsMalloc_16s(int len);
Ipp32s* ippsMalloc_32s(int len);
Ipp64s* ippsMalloc_64s(int len);
Ipp32f* ippsMalloc_32f(int len);
Ipp64f* ippsMalloc_64f(int len);
Ipp8sc* ippsMalloc_8sc(int len);
Ipp16sc* ippsMalloc_16sc(int len);
Ipp32sc* ippsMalloc_32sc(int len);
Ipp64sc* ippsMalloc_64sc(int len);
Ipp32fc* ippsMalloc_32fc(int len);
Ipp64fc* ippsMalloc_64fc(int len);
```


ippsFree

Frees memory allocated by the function `ippsMalloc`.

```
void ippsFree(void* ptr);
```

Common Functions

ippGetStatusString
Translates a status code into a message.

```
const char* ippGetStatusString(IppStatus StsCode);
```

ippGetCpuType
Returns a processor type.

```
IppCpuType ippGetCpuType (void);
```

ippGetCpuClocks
Returns a current value of the time stamp counter (TSC) register.

```
Ipp64u ippGetCpuClocks (void);
```

ippGetCpuFreqMhz
Estimates the processor operating frequency.

```
IppStatus ippGetCpuFreqMhz(int* pMhz);
```

ippGetMaxCacheSizeB
Returns maximum size of the L2 and L3 caches of the processor.

```
IppStatus ippGetMaxCacheSizeB(int* pSizeByte);
```

ippSetFlushToZero
Enables or disables flush-to-zero mode.

```
IppStatus ippSetFlushToZero(int value, unsigned int* pUMask);
```

ippSetDenormAreZeros
Enables or disables denormals-are-zero mode.

```
IppStatus ippSetDenormAreZeros(int value);
```

ippAlignPtr
Aligns a pointer to the specified number of bytes.

```
void* ippAlignPtr(void* ptr, int alignBytes);
```

ippSetNumThreads
Sets the number of threads in the multithreading environment.

```
IppStatus ippSetNumThreads(int numThr);
```

ippGetNumThreads
Returns the number of existing threads in the multithreading environment.

```
IppStatus ippGetNumThreads(int* pNumThr);
```

ippMalloc
Allocates memory aligned to 32-byte boundary.

```
void* ippMalloc(int length);
```

ippFree
Frees memory allocated by the function `ippMalloc`.

```
void ippFree(void* ptr);
```

Dispatcher Control Functions

ippStaticInit
Automatically selects the most appropriate static code.

```
IppStatus ippStaticInit(void);
```

ippStaticInitCpu
Initializes the specified version of the static code.

```
IppStatus ippStaticInitCpu(IppCpuType cpu);
```

Vector Initialization Functions

Vector Initialization Functions

Copy
Copies the contents of one vector into another.

```
IppStatus ippsCopy_1u(const Ipp8u* pSrc, int srcBitOffset, Ipp8u* pDst,
   int dstBitOffset, int len);
IppStatus ippsCopy_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsCopy_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsCopy_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len);
IppStatus ippsCopy_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCopy_64s(const Ipp64s* pSrc, Ipp64s* pDst, int len );
IppStatus ippsCopy_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len );
IppStatus ippsCopy_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len);
IppStatus ippsCopy_32sc(const Ipp32sc* pSrc, Ipp32sc* pDst, int len);
IppStatus ippsCopy_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsCopy_64sc(const Ipp64sc* pSrc, Ipp64sc* pDst, int len);
IppStatus ippsCopy_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
```

Move
Moves the contents of one vector to another vector.

```
IppStatus ippsMove_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
```

```
IppStatus ippsMove_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsMove_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len);
IppStatus ippsMove_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsMove_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len );
IppStatus ippsMove_64s(const Ipp64s* pSrc, Ipp64s* pDst, int len);
IppStatus ippsMove_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len);
IppStatus ippsMove_32sc(const Ipp32sc* pSrc, Ipp32sc* pDst, int len);
IppStatus ippsMove_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsMove_64sc(const Ipp64sc* pSrc, Ipp64sc* pDst, int len);
IppStatus ippsMove_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
```

Set

Initializes vector elements to a specified common value.

```
IppStatus ippsSet_8u(Ipp8u val, Ipp8u* pDst, int len);
IppStatus ippsSet_16s(Ipp16s val, Ipp16s* pDst, int len);
IppStatus ippsSet_16sc(Ipp16sc val, Ipp16sc* pDst, int len);
IppStatus ippsSet_32s(Ipp32s val, Ipp32s* pDst, int len);
IppStatus ippsSet_32f(Ipp32f val, Ipp32f* pDst, int len);
IppStatus ippsSet_32sc(Ipp32sc val, Ipp32sc* pDst, int len);
IppStatus ippsSet_32fc(Ipp32fc val, Ipp32fc* pDst, int len);
IppStatus ippsSet_64s(Ipp64s val, Ipp64s* pDst, int len);
IppStatus ippsSet_64f(Ipp64f val, Ipp64f* pDst, int len);
IppStatus ippsSet_64sc(Ipp64sc val, Ipp64sc* pDst, int len);
IppStatus ippsSet_64fc(Ipp64fc val, Ipp64fc* pDst, int len);
```

Zero

Initializes a vector to zero.

```
IppStatus ippsZero_8u(Ipp8u* pDst, int len);
IppStatus ippsZero_16s(Ipp16s* pDst, int len);
IppStatus ippsZero_32s(Ipp32s* pDst, int len);
IppStatus ippsZero_32f(Ipp32f* pDst, int len);
IppStatus ippsZero_64s(Ipp64s* pDst, int len);
IppStatus ippsZero_64f(Ipp64f* pDst, int len);
IppStatus ippsZero_16sc(Ipp16sc* pDst, int len);
IppStatus ippsZero_32sc(Ipp32sc* pDst, int len);
```

```
IppStatus ippsZero_32fc(Ipp32fc* pDst, int len);
IppStatus ippsZero_64sc(Ipp64sc* pDst, int len);
IppStatus ippsZero_64fc(Ipp64fc* pDst, int len);
```

Sample-Generating Functions

ToneInitAllocQ15

Allocates memory and initializes the tone generator specification structure for fixed point data.

```
IppStatus ippsToneInitAllocQ15_16s(IppToneState_16s** pToneState, Ipp16s
   magn, Ipp16s rFreqQ15, Ipp32s phaseQ15);
```

ToneFree

Frees memory allocated by the function ippsToneInitAllocQ15.

```
IppStatus ippsToneFree(IppToneState_16s* pToneState);
```

ToneGetStateSizeQ15

Computes the length of the tone generator structure.

```
IppStatus ippsToneGetStateSizeQ15_16s(int* pToneStateSize);
```

ToneInitQ15

Initializes the tone generator specification structure for fixed point data.

```
IppStatus ippsToneInitQ15_16s(IppToneState_16s* pToneState, Ipp16s magn,
   Ipp16s rFreqQ15, Ipp32s phaseQ15);
```

ToneQ15

Generates a tone with a frequency, phase, and magnitude specified in the tone generator structure.

```
IppStatus ippsToneQ15_16s(Ipp16s* pDst, int len, IppToneState_16s*
   pToneState);
```

Tone_Direct

Generates a tone with a given frequency, phase, and magnitude.

```
IppStatus ippsTone_Direct_16s(Ipp16s* pDst, int len, Ipp16s magn, float
   rFreq, float* pPhase, IppHintAlgorithm hint);
IppStatus ippsTone_Direct_16sc(Ipp16sc* pDst, int len, Ipp16s magn, float
   rFreq, float* pPhase, IppHintAlgorithm hint);
IppStatus ippsTone_Direct_32f(Ipp32f* pDst, int len, float magn, float
   rFreq, float* pPhase, IppHintAlgorithm hint);
IppStatus ippsTone_Direct_32fc(Ipp32fc* pDst, int len, float magn, float
   rFreq, float* pPhase, IppHintAlgorithm hint);
IppStatus ippsTone_Direct_64f(Ipp64f* pDst, int len, double magn, double
   rFreq, double* pPhase, IppHintAlgorithm hint);
IppStatus ippsTone_Direct_64fc(Ipp64fc* pDst, int len, double magn,
   double rFreq, double* pPhase, IppHintAlgorithm hint);
```

ToneQ15_Direct

Generates a tone with a given frequency, phase, and magnitude.

```
IppStatus ippsToneQ15_Direct_16s(Ipp16s* pDst, int len, Ipp16s magn,
   Ipp16s rFreqQ15, Ipp32s phaseQ15);
```

TriangleInitAllocQ15

Allocates memory and initializes the triangle generator specification structure for fixed point data.

```
IppStatus ippsTriangleInitAllocQ15_16s(IppTriangleState_16s**
   pTriangleState, Ipp16s magn, Ipp16s rFreqQ15, Ipp32s phaseQ15, Ipp32s
   asymQ15);
```

TriangleFree

Frees memory allocated by the function ippsTriangleInitAlloc.

```
IppStatus ippsTriangleFree(IppTriangleState_16s* pTriangleState);
```

TriangleGetStateSizeQ15

Computes the length of the triangle generator structure.

```
IppStatus ippsTriangleGetStateSizeQ15_16s(int* pTriangleStateSize);
```

TriangleInitQ15

Initializes the triangle generator specification structure for fixed point data.

```
IppStatus ippsTriangleInitQ15_16s(IppTriangleState_16s* pTriangleState,
   Ipp16s magn, Ipp16s rFreqQ15, Ipp32s phaseQ15, Ipp32s asymQ15);
```

TriangleQ15

Generates a triangle with a frequency, phase, and magnitude specified in the triangle generator structure.

```
IppStatus ippsTriangleQ15_16s(Ipp16s* pDst, int len,
   IppTriangleState_16s* pTriangleState);
```

Triangle_Direct

Generates a triangle with a given frequency, phase, and magnitude.

```
IppStatus ippsTriangle_Direct_16s(Ipp16s* pDst, int len, Ipp16s magn,
   float rFreq, float asym, float* pPhase);

IppStatus ippsTriangle_Direct_16sc(Ipp16sc* pDst, int len, Ipp16s magn,
   float rFreq, float asym, float* pPhase);

IppStatus ippsTriangle_Direct_32f(Ipp32f* pDst, int len, float magn,
   float rFreq, float asym, float* pPhase);

IppStatus ippsTriangle_Direct_32fc(Ipp32fc* pDst, int len, float magn,
   float rFreq, float asym, float* pPhase);

IppStatus ippsTriangle_Direct_64f(Ipp64f* pDst, int len, double magn,
   double rFreq, double asym, double* pPhase);

IppStatus ippsTriangle_Direct_64fc(Ipp64fc* pDst, int len, double magn,
   double rFreq, double asym, double* pPhase);
```

TriangleQ15_Direct

Generates a triangle with a given frequency, phase, and magnitude for fixed point data.

```
IppStatus ippsTriangleQ15_Direct_16s(Ipp16s* pDst, int len, Ipp16s magn,
   Ipp16s rFreqQ15, Ipp32s phaseQ15, Ipp32s asymQ15);
```

RandUniformInitAlloc

Allocates memory and initializes a noise generator with uniform distribution.

```
IppStatus ippsRandUniformInitAlloc_8u(IppsRandUniState_8u**
   pRandUniState, Ipp8u low, Ipp8u high, unsigned int seed);
IppStatus ippsRandUniformInitAlloc_16s(IppsRandUniState_16s**
   pRandUniState, Ipp16s low, Ipp16s high, unsigned int seed);
IppStatus ippsRandUniformInitAlloc_32f(IppsRandUniState_32f**
   pRandUniState, Ipp32f low, Ipp32f high, unsigned int seed);
```

RandUniformFree

Closes the uniform distribution generator state.

```
IppStatus ippsRandUniformFree_8u(IppsRandUniState_8u* pRandUniState);
IppStatus ippsRandUniformFree_16s(IppsRandUniState_16s* pRandUniState);
IppStatus ippsRandUniformFree_32f(IppsRandUniState_32f* pRandUniState);
```

RandUniformInit

Initializes a noise generator with uniform distribution.

```
IppStatus ippsRandUniformInit_16s(IppsRandUniState_16s* pRandUniState,
   Ipp16s low, Ipp16s high, unsigned int seed);
```

RandUniformGetSize

Computes the length of the uniform distribution generator structure.

```
IppStatus ippsRandUniformGetSize_16s(int* pRandUniStateSize);
```

RandUnifrom

Generates the pseudo-random samples with a uniform distribution.

```
IppStatus ippsRandUniform_8u(Ipp8u* pDst,  int len, IppsRandUniState_8u*
   pRandUniState);
IppStatus ippsRandUniform_16s(Ipp16s* pDst, int len,
   IppsRandUniState_16s* pRandUniState);
IppStatus ippsRandUniform_32f(Ipp32f* pDst, int len,
   IppsRandUniState_32f* pRandUniState);
```

RandUniform_Direct

Generates the pseudo-random samples with a uniform distribution in direct mode.

```
IppStatus ippsRandUniform_Direct_16s(Ipp16s* pDst, int len, Ipp16s low,
   Ipp16s high, unsigned int* pSeed);
IppStatus ippsRandUniform_Direct_32f(Ipp32f* pDst, int len, Ipp32f low,
   Ipp32f high, unsigned int* pSeed);
```

```
IppStatus ippsRandUniform_Direct_64f(Ipp64f* pDst, int len, Ipp64f low,
   Ipp64f high, unsigned int* pSeed);
```

RandGaussInitAlloc

Allocates memory and initializes a noise generator with Gaussian distribution.

```
IppStatus ippsRandGaussInitAlloc_8u(IppsRandGaussState_8u**
   pRandGaussState, Ipp8u mean, Ipp8u stdDev, unsigned int seed);
IppStatus ippsRandGaussInitAlloc_16s(IppsRandGaussState_16s**
   pRandGaussState, Ipp16s mean, Ipp16s stdDev, unsigned int seed);
IppStatus ippsRandGaussInitAlloc_32f(IppsRandGaussState_32f**
   pRandGaussState, Ipp32f mean, Ipp32f stdDev, unsigned int seed);
```

RandGaussFree

Closes the Gaussian distribution generator state.

```
IppStatus ippsRandGaussFree_8u(IppsRandGaussState_8u* pRandGaussState);
IppStatus ippsRandGaussFree_16s(IppsRandGaussState_16s*
   pRandGaussState);
IppStatus ippsRandGaussFree_32f(IppsRandGaussState_32f*
   pRandGaussState);
```

RandGaussGetSize

Computes the length of the Gaussian distribution generator structure.

```
IppStatus ippsRandGaussGetSize_16s(int* pRandGaussStateSize);
```

RandGaussInit

Initializes a noise generator with Gaussian distribution.

```
IppStatus ippsRandGaussInit_16s(IppsRandGaussState_16s* pRandGaussState,
   Ipp16s mean, Ipp16s stdDev, unsigned int seed);
```

RandGauss

Generates the pseudo-random samples with a Gaussian distribution.

```
IppStatus ippsRandGauss_8u(Ipp8u* pDst,  int len, IppsRandGaussState_8u*
   pRandGaussState);
IppStatus ippsRandGauss_16s(Ipp16s* pDst, int len,
   IppsRandGaussState_16s* pRandGaussState);
IppStatus ippsRandGauss_32f(Ipp32f* pDst, int len,
   IppsRandGaussState_32f* pRandGaussState);
```

RandGauss_Direct

Generates pseudo-random samples with a Gaussian distribution in the direct mode.

```
IppStatus ippsRandGauss_Direct_16s(Ipp16s* pDst, int len, Ipp16s mean,
   Ipp16s stdev, unsigned int* pSeed);
IppStatus ippsRandGauss_Direct_32f(Ipp32f* pDst, int len, Ipp32f mean,
   Ipp32f stdev, unsigned int* pSeed);
IppStatus ippsRandGauss_Direct_64f(Ipp64f* pDst, int len, Ipp64f mean,
   Ipp64f stdev, unsigned int* pSeed);
```

VectorJaehne

Creates a Jaehne vector.

```
IppStatus ippsVectorJaehne_8u(Ipp8u* pDst, int len, Ipp8u magn);
IppStatus ippsVectorJaehne_8s(Ipp8s* pDst, int len, Ipp8s magn);
IppStatus ippsVectorJaehne_16u(Ipp16u* pDst, int len, Ipp16u magn);
IppStatus ippsVectorJaehne_16s(Ipp16s* pDst, int len, Ipp16s magn);
IppStatus ippsVectorJaehne_32u(Ipp32u* pDst, int len, Ipp32u magn);
IppStatus ippsVectorJaehne_32s(Ipp32s* pDst, int len, Ipp32s magn);
IppStatus ippsVectorJaehne_32f(Ipp32f* pDst, int len, Ipp32f magn);
IppStatus ippsVectorJaehne_64f(Ipp64f* pDst, int len, Ipp64f magn);
```

VectorRamp

Creates a ramp vector.

```
IppStatus ippsVectorRamp_8u(Ipp8u* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_8s(Ipp8s* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_16u(Ipp16u* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_16s(Ipp16s* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_32u(Ipp32u* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_32s(Ipp32s* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_32f(Ipp32f* pDst, int len, float offset, float
   slope);
IppStatus ippsVectorRamp_64f(Ipp64f* pDst, int len, float offset, float
   slope);
```

Essential Vector Functions

Logical and Shift Functions

AndC

Computes the bitwise AND of a scalar value and each element of a vector.

```
IppStatus ippsAndC_8u(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int len);
```

```
IppStatus ippsAndC_16u(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst, int len);
IppStatus ippsAndC_32u(const Ipp32u* pSrc, Ipp32u val, Ipp32u* pDst, int len);
IppStatus ippsAndC_8u_I(Ipp8u val, Ipp8u* pSrcDst, int len);
IppStatus ippsAndC_16u_I(Ipp16u val, Ipp16u* pSrcDst, int len);
IppStatus ippsAndC_32u_I(Ipp32u val, Ipp32u* pSrcDst, int len);
```

And

Computes the bitwise AND of two vectors.

```
IppStatus ippsAnd_8u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u* pDst, int
   len);
IppStatus ippsAnd_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, Ipp16u* pDst,
   int len);
IppStatus ippsAnd_32u(const Ipp32u* pSrc1, const Ipp32u* pSrc2, Ipp32u* pDst,
   int len);
IppStatus ippsAnd_8u_I(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len);
IppStatus ippsAnd_16u_I(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len);
IppStatus ippsAnd_32u_I(const Ipp32u* pSrc, Ipp32u* pSrcDst, int len);
```

OrC

Computes the bitwise OR of a scalar value and each element of a vector.

```
IppStatus ippsOrC_8u(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int len);
IppStatus ippsOrC_16u(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst, int len);
IppStatus ippsOrC_32u(const Ipp32u* pSrc, Ipp32u val, Ipp32u* pDst, int len);
IppStatus ippsOrC_8u_I(Ipp8u val, Ipp8u* pSrcDst, int len);
IppStatus ippsOrC_16u_I(Ipp16u val, Ipp16u* pSrcDst, int len);
IppStatus ippsOrC_32u_I(Ipp16u val, Ipp32u* pSrcDst, int len);
```

Or

Computes the bitwise OR of two vectors.

```
IppStatus ippsOr_8u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u* pDst, int
   len);
IppStatus ippsOr_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, Ipp16u* pDst,
   int len);
IppStatus ippsOr_32u(const Ipp32u* pSrc1, const Ipp32u* pSrc2, Ipp32u* pDst,
   int len);
IppStatus ippsOr_8u_I(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len);
IppStatus ippsOr_16u_I(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len);
IppStatus ippsOr_32u_I(const Ipp32u* pSrc, Ipp32u* pSrcDst, int len);
```

XorC

Computes the bitwise XOR of a scalar value and each element of a vector.

```
IppStatus ippsXorC_8u(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int len);
IppStatus ippsXorC_16u(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst, int len);
IppStatus ippsXorC_32u(const Ipp32u* pSrc, Ipp32u val, Ipp32u* pDst, int len);
IppStatus ippsXorC_8u_I(Ipp8u val, Ipp8u* pSrcDst, int len);
IppStatus ippsXorC_16u_I(Ipp16u val, Ipp16u* pSrcDst, int len);
IppStatus ippsXorC_32u_I(Ipp32u val, Ipp32u* pSrcDst, int len);
```

Xor

Computes the bitwise XOR of two vectors.

```
IppStatus ippsXor_8u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u* pDst, int
   len);
IppStatus ippsXor_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, Ipp16u* pDst,
   int len);
IppStatus ippsXor_32u(const Ipp32u* pSrc1, const Ipp32u* pSrc2, Ipp32u* pDst,
   int len);
IppStatus ippsXor_8u_I(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len);
IppStatus ippsXor_16u_I(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len);
IppStatus ippsXor_32u_I(const Ipp32u* pSrc, Ipp32u* pSrcDst, int len);
```

Not

Computes the bitwise NOT of the vector elements.

```
IppStatus ippsNot_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsNot_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len);
IppStatus ippsNot_32u(const Ipp32u* pSrc, Ipp32u* pDst, int len);
IppStatus ippsNot_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsNot_16u_I(Ipp16u* pSrcDst, int len);
IppStatus ippsNot_32u_I(Ipp32u* pSrcDst, int len);
```

LShiftC

Shifts bits in vector elements to the left.

```
IppStatus ippsLShiftC_8u(const Ipp8u* pSrc, int val, Ipp8u* pDst, int len);
IppStatus ippsLShiftC_16s(const Ipp16s* pSrc, int val, Ipp16s* pDst, int len);
IppStatus ippsLShiftC_16u(const Ipp16u* pSrc, int val, Ipp16u* pDst, int len);
IppStatus ippsLShiftC_32s(const Ipp32s* pSrc, int val, Ipp32s* pDst, int len);
IppStatus ippsLShiftC_8u_I(int val, Ipp8u* pSrcDst, int len);
IppStatus ippsLShiftC_16u_I(int val, Ipp16u* pSrcDst, int len);
```

```
IppStatus ippsLShiftC_16s_I(int val, Ipp16s* pSrcDst, int len);
IppStatus ippsLShiftC_32s_I(int val, Ipp32s* pSrcDst, int len);
```

RShiftC

Shifts bits in vector elements to the right.

```
IppStatus ippsRShiftC_8u(const Ipp8u* pSrc, int val, Ipp8u* pDst, int len);
IppStatus ippsRShiftC_16s(const Ipp16s* pSrc, int val, Ipp16s* pDst, int len);
IppStatus ippsRShiftC_16u(const Ipp16u* pSrc, int val, Ipp16u* pDst, int len);
IppStatus ippsRShiftC_32s(const Ipp32s* pSrc, int val, Ipp32s* pDst, int len);
IppStatus ippsRShiftC_8u_I(int val, Ipp8u* pSrcDst, int len);
IppStatus ippsRShiftC_16u_I(int val, Ipp16u* pSrcDst, int len);
IppStatus ippsRShiftC_16s_I(int val, Ipp16s* pSrcDst, int len);
IppStatus ippsRShiftC_32s_I(int val, Ipp32s* pSrcDst, int len);
```

Arithmetic Functions

AddC

Adds a constant value to each element of a vector.

```
IppStatus ippsAddC_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst, int len);
IppStatus ippsAddC_64f(const Ipp64f* pSrc, Ipp64f val, Ipp64f* pDst, int len);
IppStatus ippsAddC_32fc(const Ipp32fc* pSrc, Ipp32fc val,Ipp32fc* pDst, int
    len);
IppStatus ippsAddC_64fc(const Ipp64fc* pSrc, Ipp64fc val, Ipp64fc* pDst, int
    len);
IppStatus ippsAddC_16s_I(Ipp16s val, Ipp16s* pSrcDst, int len);
IppStatus ippsAddC_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
IppStatus ippsAddC_64f_I(Ipp64f val, Ipp64f* pSrcDst, int len);
IppStatus ippsAddC_32fc_I(Ipp32fc val, Ipp32fc* pSrcDst, int len);
IppStatus ippsAddC_64fc_I(Ipp64fc val, Ipp64fc* pSrcDst, int len);
IppStatus ippsAddC_8u_Sfs(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int len,
    int scaleFactor);
IppStatus ippsAddC_16s_Sfs(const Ipp16s* pSrc, Ipp16s val, Ipp16s* pDst, int
    len, int scaleFactor);
IppStatus ippsAddC_16u_Sfs(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst, int
    len, int scaleFactor);
IppStatus ippsAddC_32s_Sfs(const Ipp32s* pSrc, Ipp32s val, Ipp32s* pDst, int
    len, int scaleFactor);
IppStatus ippsAddC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc val, Ipp16sc* pDst,
    int len, int scaleFactor);
```

```
IppStatus ippsAddC_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc val, Ipp32sc*
   pDst, int len, int scaleFactor);
IppStatus ippsAddC_8u_ISfs(Ipp8u val, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAddC_16u_ISfs(Ipp16u val, Ipp16u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAddC_16s_ISfs(Ipp16s val, Ipp16s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAddC_32s_ISfs(Ipp32s val, Ipp32s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAddC_16sc_ISfs(Ipp16sc val, Ipp16sc* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAddC_32sc_ISfs(Ipp32sc val, Ipp32sc* pSrcDst, int len, int
   scaleFactor);
```

Add

Adds the elements of two vectors.

```
IppStatus ippsAdd_16s(const Ipp16s* pSrc1, const Ipp16s* pSrc2, Ipp16s*
   pDst, int len);
IppStatus ippsAdd_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, Ipp16u*
   pDst, int len);
IppStatus ippsAdd_32u(const Ipp32u* pSrc1, const Ipp32u* pSrc2, Ipp32u*
   pDst, int len);
IppStatus ippsAdd_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2, Ipp32f*
   pDst, int len);
IppStatus ippsAdd_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2, Ipp64f*
   pDst, int len);
IppStatus ippsAdd_32fc(const Ipp32fc* pSrc1, const Ipp32fc* pSrc2,
   Ipp32fc* pDst, int len);
IppStatus ippsAdd_64fc(const Ipp64fc* pSrc1, const Ipp64fc* pSrc2,
   Ipp64fc* pDst, int len);
IppStatus ippsAdd_8u16u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp16u*
   pDst, int len);
IppStatus ippsAdd_16s32f(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsAdd_16s_I(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len);
IppStatus ippsAdd_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len);
IppStatus ippsAdd_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len);
IppStatus ippsAdd_32fc_I(const Ipp32fc* pSrc, Ipp32fc* pSrcDst, int len);
IppStatus ippsAdd_64fc_I(const Ipp64fc* pSrc, Ipp64fc* pSrcDst, int len);
IppStatus ippsAdd_16s32s_I(const Ipp16s* pSrc, Ipp32s* pSrcDst, int len);
```

```
IppStatus ippsAdd_8u_Sfs(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u*
   pDst, int len, int scaleFactor);
IppStatus ippsAdd_16u_Sfs(const Ipp16u* pSrc1, const Ipp16u* pSrc2,
   Ipp16u* pDst, int len, int scaleFactor);
IppStatus ippsAdd_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsAdd_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
IppStatus ippsAdd_16sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc* pSrc2,
   Ipp16sc* pDst, int len, int scaleFactor);
IppStatus ippsAdd_32sc_Sfs(const Ipp32sc* pSrc1, const Ipp32sc* pSrc2,
   Ipp32sc* pDst, int len, int scaleFactor);
IppStatus ippsAdd_8u_ISfs(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsAdd_16u_ISfs(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsAdd_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsAdd_32s_ISfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsAdd_16sc_ISfs(const Ipp16sc* pSrc, Ipp16sc* pSrcDst, int
   len, int scaleFactor);
IppStatus ippsAdd_32sc_ISfs(const Ipp32sc* pSrc, Ipp32sc* pSrcDst, int
   len, int scaleFactor)
```

AddProduct

Adds product of two vectors to the accumulator vector.

```
IppStatus ippsAddProduct_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pSrcDst, int len);
IppStatus ippsAddProduct_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pSrcDst, int len);
IppStatus ippsAddProduct_32fc(const Ipp32fc* pSrc1, const Ipp32fc*
   pSrc2, Ipp32fc* pSrcDst, int len);
IppStatus ippsAddProduct_64fc(const Ipp64fc* pSrc1, const Ipp64fc*
   pSrc2, Ipp64fc* pSrcDst, int len);
IppStatus ippsAddProduct_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pSrcDst, int len, int scaleFactor);
IppStatus ippsAddProduct_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s*
   pSrc2, Ipp32s* pSrcDst, int len, int scaleFactor);
IppStatus ippsAddProduct_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp32s* pSrcDst, int len, int scaleFactor);
```

AddProductC

Adds product of a vector and a constant to the accumulator vector.

```
IppStatus ippsAddProductC_32f(const Ipp32f* pSrc, const Ipp32f val,
   Ipp32f* pSrcDst, int len);
```

MulC

Multiplies each elements of a vector by a constant value.

```
IppStatus ippsMulC_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst, int
   len);
IppStatus ippsMulC_64f(const Ipp64f* pSrc, Ipp64f val, Ipp64f* pDst, int
   len);
IppStatus ippsMulC_32fc(const Ipp32fc* pSrc, Ipp32fc val, Ipp32fc* pDst,
   int len);
IppStatus ippsMulC_64fc(const Ipp64fc* pSrc, Ipp64fc val, Ipp64fc* pDst,
   int len);
IppStatus ippsMulC_16s_I(Ipp16s val, Ipp16s* pSrcDst, int len);
IppStatus ippsMulC_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
IppStatus ippsMulC_64f_I(Ipp64f val, Ipp64f* pSrcDst, int len);
IppStatus ippsMulC_32fc_I(Ipp32fc val, Ipp32fc* pSrcDst, int len);
IppStatus ippsMulC_64fc_I(Ipp64fc val, Ipp64fc* pSrcDst, int len);
IppStatus ippsMulC_8u_Sfs(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int
   len, int scaleFactor);
IppStatus ippsMulC_16s_Sfs(const Ipp16s* pSrc, Ipp16s val, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsMulC_16u_Sfs(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst,
   int len, int scaleFactor);
IppStatus ippsMulC_32s_Sfs(const Ipp32s* pSrc, Ipp32s val, Ipp32s* pDst,
   int len, int scaleFactor);
IppStatus ippsMulC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc val, Ipp16sc*
   pDst, int len, int scaleFactor);
IppStatus ippsMulC_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc val, Ipp32sc*
   pDst, int len, int scaleFactor);
IppStatus ippsMulC_8u_ISfs(Ipp8u val, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsMulC_16u_ISfs(Ipp16u val, Ipp16u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsMulC_16s_ISfs(Ipp16s val, Ipp16s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsMulC_32s_ISfs(Ipp32s val, Ipp32s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsMulC_16sc_ISfs(Ipp16sc val, Ipp16sc* pSrcDst, int len, int
   scaleFactor);
```

```
IppStatus ippsMulC_32sc_ISfs(Ipp32sc val, Ipp32sc* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsMulC_32f16s_Sfs(const Ipp32f* pSrc, Ipp32f val, Ipp16s*
   pDst, int len, int scaleFactor);
IppStatus ippsMulC_Low_32f16s(const Ipp32f* pSrc, Ipp32f val, Ipp16s*
   pDst, int len);
```

Mul

Multiplies the elements of two vectors.

```
IppStatus ippsMul_16s(const Ipp16s* pSrc1, const Ipp16s* pSrc2, Ipp16s*
   pDst, int len);
IppStatus ippsMul_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2, Ipp32f*
   pDst, int len);
IppStatus ippsMul_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2, Ipp64f*
   pDst, int len);
IppStatus ippsMul_32fc(const Ipp32fc* pSrc1, const Ipp32fc* pSrc2,
   Ipp32fc* pDst, int len);
IppStatus ippsMul_64fc(const Ipp64fc* pSrc1, const Ipp64fc* pSrc2,
   Ipp64fc* pDst, int len);
IppStatus ippsMul_8u16u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp16u*
   pDst, int len);
IppStatus ippsMul_16s32f(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsMul_32f32fc(const Ipp32f* pSrc1, const Ipp32fc* pSrc2,
   Ipp32fc* pDst, int len);


IppStatus ippsMul_16s_I(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len);
IppStatus ippsMul_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len);
IppStatus ippsMul_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len);
IppStatus ippsMul_32fc_I(const Ipp32fc* pSrc, Ipp32fc* pSrcDst, int len);
IppStatus ippsMul_64fc_I(const Ipp64fc* pSrc, Ipp64fc* pSrcDst, int len);
IppStatus ippsMul_32f32fc_I(const Ipp32f* pSrc, Ipp32fc* pSrcDst, int
   len);


IppStatus ippsMul_8u_Sfs(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u*
   pDst, int len, int scaleFactor);
IppStatus ippsMul_16u_Sfs(const Ipp16u* pSrc1, const Ipp16u* pSrc2,
   Ipp16u* pDst, int len, int scaleFactor);
IppStatus ippsMul_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsMul_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
```

```
IppStatus ippsMul_16sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc* pSrc2,
   Ipp16sc* pDst, int len, int scaleFactor);
IppStatus ippsMul_32sc_Sfs(const Ipp32sc* pSrc1, const Ipp32sc* pSrc2,
   Ipp32sc* pDst, int len, int scaleFactor);
IppStatus ippsMul_16u16s_Sfs(const Ipp16u* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsMul_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
IppStatus ippsMul_32s32sc_Sfs(const Ipp32s* pSrc1, const Ipp32sc* pSrc2,
   Ipp32sc* pDst, int len, int scaleFactor);


IppStatus ippsMul_8u_ISfs(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus, ippsMul_16u_ISfs(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsMul_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsMul_32s_ISfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsMul_16sc_ISfs(const Ipp16sc* pSrc, Ipp16sc* pSrcDst, int
   len, int scaleFactor);
IppStatus ippsMul_32sc_ISfs(const Ipp32sc* pSrc, Ipp32sc* pSrcDst, int
   len, int scaleFactor);
IppStatus ippsMul_32s32sc_ISfs(const Ipp32s* pSrc, Ipp32sc* pSrcDst, int
   len, int scaleFactor);


IppStatus ippsMul_Low_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
```

SubC

Subtracts a constant value from each element of a vector.

```
IppStatus ippsSubC_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst, int
   len);
IppStatus ippsSubC_32fc(const Ipp32fc* pSrc, Ipp32fc val, Ipp32fc* pDst,
   int len);
IppStatus ippsSubC_64f(const Ipp64f* pSrc, Ipp64f val, Ipp64f* pDst, int
   len);
IppStatus ippsSubC_64fc(const Ipp64fc* pSrc, Ipp64fc val, Ipp64fc* pDst,
   int len);
IppStatus ippsSubC_16s_I(Ipp16s val, Ipp16s* pSrcDst, int len);
IppStatus ippsSubC_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
IppStatus ippsSubC_64f_I(Ipp64f val, Ipp64f* pSrcDst, int len);
```

```
IppStatus ippsSubC_32fc_I(Ipp32fc val, Ipp32fc* pSrcDst, int len);
IppStatus ippsSubC_64fc_I(Ipp64fc val, Ipp64fc* pSrcDst, int len);
IppStatus ippsSubC_8u_Sfs(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int
   len, int scaleFactor);
IppStatus ippsSubC_16u_Sfs(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst,
   int len, int scaleFactor);
IppStatus ippsSubC_16s_Sfs(const Ipp16s* pSrc, Ipp16s val, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsSubC_32s_Sfs(const Ipp32s* pSrc, Ipp32s val, Ipp32s* pDst,
   int len, int scaleFactor);
IppStatus ippsSubC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc val, Ipp16sc*
   pDst, int len, int scaleFactor);
IppStatus ippsSubC_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc val, Ipp32sc*
   pDst, int len, int scaleFactor);
IppStatus ippsSubC_8u_ISfs(Ipp8u val, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubC_16u_ISfs(Ipp16u val, Ipp16u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubC_16s_ISfs(Ipp16s val, Ipp16s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubC_32s_ISfs(Ipp32s val, Ipp32s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubC_16sc_ISfs(Ipp16sc val, Ipp16sc* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubC_32sc_ISfs(Ipp32sc val, Ipp32sc* pSrcDst, int len, int
   scaleFactor);
```

SubCRev

Subtracts each element of a vector from a constant value.

```
IppStatus ippsSubCRev_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst,
   int len);
IppStatus ippsSubCRev_64f(const Ipp64f* pSrc, Ipp64f val, Ipp64f* pDst,
   int len);
IppStatus ippsSubCRev_32fc(const Ipp32fc* pSrc, Ipp32fc val, Ipp32fc*
   pDst, int len);
IppStatus ippsSubCRev_64fc(const Ipp64fc* pSrc, Ipp64fc val, Ipp64fc*
   pDst, int len);
IppStatus ippsSubCRev_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
IppStatus ippsSubCRev_64f_I(Ipp64f val, Ipp64f* pSrcDst, int len);
IppStatus ippsSubCRev_32fc_I(Ipp32fc val, Ipp32fc* pSrcDst, int len);
IppStatus ippsSubCRev_64fc_I(Ipp64fc val, Ipp64fc* pSrcDst, int len);
```

```
IppStatus ippsSubCRev_8u_Sfs(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst,
   int len, int scaleFactor);
IppStatus ippsSubCRev_16u_ISfs(Ipp16u val, Ipp16u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubCRev_16u_Sfs(const Ipp16u* pSrc, Ipp16u val, Ipp16u*
   pDst, int len, int scaleFactor);
IppStatus ippsSubCRev_16s_Sfs(const Ipp16s* pSrc, Ipp16s val, Ipp16s*
   pDst, int len, int scaleFactor);
IppStatus ippsSubCRev_32s_Sfs(const Ipp32s* pSrc, Ipp32s val, Ipp32s*
   pDst, int len, int scaleFactor);
IppStatus ippsSubCRev_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc val,
   Ipp16sc* pDst, int len, int scaleFactor);
IppStatus ippsSubCRev_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc val,
   Ipp32sc* pDst, int len, int scaleFactor);
IppStatus ippsSubCRev_8u_ISfs(Ipp8u val, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubCRev_16u_ISfs(Ipp16u val, Ipp16u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubCRev_16s_ISfs(Ipp16s val, Ipp16s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubCRev_32s_ISfs(Ipp32s val, Ipp32s* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSubCRev_16sc_ISfs(Ipp16sc val, Ipp16sc* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsSubCRev_32sc_ISfs(Ipp32sc val, Ipp32sc* pSrcDst, int len,
   int scaleFactor);
```

Sub

Subtracts the elements of two vectors.

```
IppStatus ippsSub_16s(const Ipp16s* pSrc1, const Ipp16s* pSrc2, Ipp16s*
   pDst, int len);
IppStatus ippsSub_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2, Ipp32f*
   pDst, int len);
IppStatus ippsSub_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2, Ipp64f*
   pDst, int len);
IppStatus ippsSub_32fc(const Ipp32fc* pSrc1, const Ipp32fc* pSrc2,
   Ipp32fc* pDst, int len);
IppStatus ippsSub_64fc(const Ipp64fc* pSrc1, const Ipp64fc* pSrc2,
   Ipp64fc* pDst, int len);
IppStatus ippsSub_16s32f(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsSub_16s_I(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len);
IppStatus ippsSub_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len);
```

```
IppStatus ippsSub_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len);
IppStatus ippsSub_32fc_I(const Ipp32fc* pSrc, Ipp32fc* pSrcDst, int len);
IppStatus ippsSub_64fc_I(const Ipp64fc* pSrc, Ipp64fc* pSrcDst, int len);
IppStatus ippsSub_8u_Sfs(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u*
   pDst, int len, int scaleFactor);
IppStatus ippsSub_16u_Sfs(const Ipp16u* pSrc1, const Ipp16u* pSrc2,
   Ipp16u* pDst, int len, int scaleFactor);
IppStatus ippsSub_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsSub_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
IppStatus ippsSub_16sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc* pSrc2,
   Ipp16sc* pDst, int len, int scaleFactor);
IppStatus ippsSub_32sc_Sfs(const Ipp32sc* pSrc1, const Ipp32sc* pSrc2,
   Ipp32sc* pDst, int len, int scaleFactor);
IppStatus ippsSub_8u_ISfs(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSub_16u_ISfs(const Ipp16u* pSrc, Ipp16u* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsSub_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsSub_32s_ISfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int len,
   int scaleFactor);
IppStatus ippsSub_16sc_ISfs(const Ipp16sc* pSrc, Ipp16sc* pSrcDst, int
   len, int scaleFactor);
IppStatus ippsSub_32sc_ISfs(const Ipp32sc* pSrc, Ipp32sc* pSrcDst, int
   len, int scaleFactor);
```

DivC

Divides each element of a vector by a constant value.

```
IppStatus ippsDivC_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst, int
   len);
IppStatus ippsDivC_64f(const Ipp64f* pSrc, Ipp64f val, Ipp64f* pDst, int
   len);
IppStatus ippsDivC_32fc(const Ipp32fc* pSrc, Ipp32fc val, Ipp32fc* pDst,
   int len);
IppStatus ippsDivC_64fc(const Ipp64fc* pSrc, Ipp64fc val, Ipp64fc* pDst,
   int len);
IppStatus ippsDivC_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
IppStatus ippsDivC_64f_I(Ipp64f val, Ipp64f* pSrcDst, int len);
IppStatus ippsDivC_32fc_I(Ipp32fc val, Ipp32fc* pSrcDst, int len);
IppStatus ippsDivC_64fc_I(Ipp64fc val, Ipp64fc* pSrcDst, int len);
```

```
IppStatus ippsDivC_8u_Sfs(const Ipp8u* pSrc, Ipp8u val, Ipp8u* pDst, int
   len, int ScaleFactor);
IppStatus ippsDivC_16s_Sfs(const Ipp16s* pSrc, Ipp16s val, Ipp16s* pDst,
   int len, int ScaleFactor);
IppStatus ippsDivC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc val, Ipp16sc*
   pDst, int len, int ScaleFactor);
IppStatus ippsDivC_8u_ISfs(Ipp8u val, Ipp8u* pSrcDst, int len, int
   ScaleFactor);
IppStatus ippsDivC_16s_ISfs(Ipp16s val, Ipp16s* pSrcDst, int len, int
   ScaleFactor);
IppStatus ippsDivC_16sc_ISfs(Ipp16sc val, Ipp16sc* pSrcDst, int len,  int
   ScaleFactor);
```

DivCRev

Divides a constant value by each element of a vector.

```
IppStatus ippsDivCRev_16u(const Ipp16u* pSrc, Ipp16u val, Ipp16u* pDst,
   int len);
IppStatus ippsDivCRev_32f(const Ipp32f* pSrc, Ipp32f val, Ipp32f* pDst,
   int len);
IppStatus ippsDivCRev_16u_I(Ipp16u val, Ipp16u* pSrcDst, int len);
IppStatus ippsDivCRev_32f_I(Ipp32f val, Ipp32f* pSrcDst, int len);
```

Div

Divides the elements of two vectors.

```
IppStatus ippsDiv_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2, Ipp32f*
   pDst, int len);
IppStatus ippsDiv_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2, Ipp64f*
   pDst, int len);
IppStatus ippsDiv_32fc(const Ipp32fc* pSrc1, const Ipp32fc* pSrc2,
   Ipp32fc* pDst, int len);
IppStatus ippsDiv_64fc(const Ipp64fc* pSrc1, const Ipp64fc* pSrc2,
   Ipp64fc* pDst, int len);
IppStatus ippsDiv_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len);
IppStatus ippsDiv_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len);
IppStatus ippsDiv_32fc_I(const Ipp32fc* pSrc, Ipp32fc* pSrcDst, int len);
IppStatus ippsDiv_64fc_I(const Ipp64fc* pSrc, Ipp64fc* pSrcDst, int len);
IppStatus ippsDiv_8u_Sfs(const Ipp8u* pSrc1, const Ipp8u* pSrc2, Ipp8u*
   pDst, int len, int scaleFactor);
IppStatus ippsDiv_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsDiv_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   Ipp32s* pDst, int len, int scaleFactor);
```

```
IppStatus ippsDiv_16sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc* pSrc2,
   Ipp16sc* pDst, int len, int scaleFactor);
IppStatus ippsDiv_32s16s_Sfs(const Ipp16s* pSrc1, const Ipp32s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsDiv_8u_ISfs(const Ipp8u* pSrc, Ipp8u* pSrcDst, int len, int
   ScaleFactor);
IppStatus ippsDiv_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int len,
   int ScaleFactor);
IppStatus ippsDiv_16sc_ISfs(const Ipp16sc* pSrc, Ipp16sc* pSrcDst, int
   len, int ScaleFactor);
IppStatus ippsDiv_32s_ISfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int len,
   int ScaleFactor);
```

Abs

Computes absolute values of vector elements.

```
IppStatus ippsAbs_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsAbs_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len);
IppStatus ippsAbs_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAbs_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAbs_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsAbs_32s_I(Ipp32s* pSrcDst, int len);
IppStatus ippsAbs_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsAbs_64f_I(Ipp64f* pSrcDst, int len);
```

Sqr

Computes a square of each element of a vector.

```
IppStatus ippsSqr_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSqr_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsSqr_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsSqr_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
IppStatus ippsSqr_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsSqr_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsSqr_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsSqr_64fc_I(Ipp64fc* pSrcDst, int len);
IppStatus ippsSqr_8u_Sfs(const Ipp8u* pSrc, Ipp8u* pDst, int len,
   int scaleFactor);
IppStatus ippsSqr_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
```

```
IppStatus ippsSqr_16u_Sfs(const Ipp16u* pSrc, Ipp16u* pDst, int len,
   int scaleFactor);
IppStatus ippsSqr_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int len,
   int scaleFactor);
IppStatus ippsSqr_8u_ISfs(Ipp8u* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqr_16s_ISfs(Ipp16s* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqr_16u_ISfs(Ipp16u* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqr_16sc_ISfs(Ipp16sc* pSrcDst, int len, int scaleFactor);
```

Sqrt

Computes a square root of each element of a vector.

```
IppStatus ippsSqrt_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSqrt_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsSqrt_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsSqrt_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
IppStatus ippsSqrt_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsSqrt_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsSqrt_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsSqrt_64fc_I(Ipp64fc* pSrcDst, int len);
IppStatus ippsSqrt_8u_Sfs(const Ipp8u* pSrc, Ipp8u* pDst, int len, int
   scaleFactor);
IppStatus ippsSqrt_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len, int
   scaleFactor);
IppStatus ippsSqrt_16u_Sfs(const Ipp16u* pSrc, Ipp16u* pDst, int len, int
   scaleFactor);
IppStatus ippsSqrt_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int len,
   int scaleFactor);
IppStatus ippsSqrt_64s_Sfs(const Ipp64s* pSrc, Ipp64s* pDst, int len, int
   scaleFactor);
IppStatus ippsSqrt_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
IppStatus ippsSqrt_64s16s_Sfs(const Ipp64s* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
IppStatus ippsSqrt_8u_ISfs(Ipp8u* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqrt_16s_ISfs(Ipp16s* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqrt_16u_ISfs(Ipp16u* pSrcDst, int len, int scaleFactor);
IppStatus ippsSqrt_16sc_ISfs(Ipp16sc* pSrcDst, int len, int
   scaleFactor);
IppStatus ippsSqrt_64s_ISfs(Ipp64s* pSrcDst, int len, int scaleFactor);
```

Cubrt

Computes cube root of each element of a vector.

```
IppStatus ippsCubrt_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCubrt_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
```

Exp

Computes e to the power of each element of a vector.

```
IppStatus ippsExp_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsExp_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsExp_32f64f(const Ipp32f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsExp_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsExp_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsExp_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len, int
   scaleFactor);
IppStatus ippsExp_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int len, int
   scaleFactor);
IppStatus ippsExp_64s_Sfs(const Ipp64s* pSrc, Ipp64s* pDst, int len, int
   scaleFactor);
IppStatus ippsExp_16s_ISfs(Ipp16s* pSrcDst, int len, int scaleFactor);
IppStatus ippsExp_32s_ISfs(Ipp32s* pSrcDst, int len, int scaleFactor);
IppStatus ippsExp_64s_ISfs(Ipp64s* pSrcDst, int len, int scaleFactor);
```

Ln

Computes the natural logarithm of each element of a vector.

```
IppStatus ippsLn_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLn_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsLn_64f32f(const Ipp64f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLn_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsLn_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsLn_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len, int
   scaleFactor);
IppStatus ippsLn_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int len, int
   scaleFactor);
IppStatus ippsLn_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
IppStatus ippsLn_16s_ISfs(Ipp16s* pSrcDst, int len, int scaleFactor);
IppStatus ippsLn_32s_ISfs(Ipp32s* pSrcDst, int len, int scaleFactor);
```

10Log10

Computes the decimal logarithm of each element of a vector and multiplies it by 10.

```
IppStatus ipps10Log10_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int len,
   int scaleFactor);
IppStatus ipps10Log10_32s_ISfs(Ipp32s* pSrcDst, int len, int
   scaleFactor);
```

SumLn

Sums natural logarithms of each element of a vector.

```
IppStatus ippsSumLn_32f(const Ipp32f* pSrc, int len, Ipp32f* pSum);
IppStatus ippsSumLn_64f(const Ipp64f* pSrc, int len, Ipp64f* pSum);
IppStatus ippsSumLn_32f64f(const Ipp32f* pSrc, int len, Ipp64f* pSum);
IppStatus ippsSumLn_16s32f(const Ipp16s* pSrc, int len, Ipp32f* pSum);
```

Arctan

Computes the inverse tangent of each element of a vector.

```
IppStatus ippsArctan_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsArctan_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsArctan_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsArctan_64f_I(Ipp64f* pSrcDst, int len);
```

Normalize

Normalizes elements of a real or complex vector using offset and division operations.

```
IppStatus ippsNormalize_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f vsub, Ipp32f vdiv);
IppStatus ippsNormalize_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   Ipp64f vsub, Ipp64f vdiv);
IppStatus ippsNormalize_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   Ipp32fc vsub, Ipp32f vdiv);
IppStatus ippsNormalize_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len,
   Ipp64fc vsub, Ipp64f vdiv);
IppStatus ippsNormalize_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, Ipp16s vsub, int vdiv, int scaleFactor);
IppStatus ippsNormalize_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, Ipp16sc vsub, int vdiv, int scaleFactor);
```

Conversion Functions

SortAscend, SortDescend

Sorts all elements of a vector.

```
IppStatus ippsSortAscend_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsSortAscend_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsSortAscend_32s_I(Ipp32s* pSrcDst, int len);
IppStatus ippsSortAscend_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsSortAscend_64f_I(Ipp64f* pSrcDst, int len);
```

55

```
IppStatus ippsSortDescend_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsSortDescend_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsSortDescend_32s_I(Ipp32s* pSrcDst, int len);
IppStatus ippsSortDescend_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsSortDescend_64f_I(Ipp64f* pSrcDst, int len);
```

SwapBytes

Reverses the byte order of a vector.

```
IppStatus ippsSwapBytes_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len);
IppStatus ippsSwapBytes_24u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsSwapBytes_32u(const Ipp32u* pSrc, Ipp32u* pDst, int len);
IppStatus ippsSwapBytes_16u_I(Ipp16u* pSrcDst, int len);
IppStatus ippsSwapBytes_24u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsSwapBytes_32u_I(Ipp32u* pSrcDst, int len);
```

Convert

Converts the data type of a vector and stores the results in a second vector.

```
IppStatus ippsConvert_8s16s(const Ipp8s* pSrc,Ipp16s* pDst, int len);
IppStatus ippsConvert_8s32f(const Ipp8s* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_8u32f(const Ipp8u* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_16s32s(const Ipp16s* pSrc, Ipp32s* pDst, int len);
IppStatus ippsConvert_16s32f(const Ipp16s* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_16u32f(const Ipp16u* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_32s16s(const Ipp32s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsConvert_32s32f(const Ipp32s* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_32s64f(const Ipp32s* pSrc, Ipp64f* pDst, int len);
```

```
IppStatus ippsConvert_32f64f(const Ipp32f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsConvert_64f32f(const Ipp64f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_16s32f_Sfs(const Ipp16s* pSrc, Ipp32f* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_16s64f_Sfs(const Ipp16s* pSrc, Ipp64f* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32s32f_Sfs(const Ipp32s* pSrc, Ipp32f* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32s64f_Sfs(const Ipp32s* pSrc, Ipp64f* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32f8s_Sfs(const Ipp32f* pSrc, Ipp8s* pDst, int len,
    IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_32f8u_Sfs(const Ipp32f* pSrc, Ipp8u* pDst, int len,
    IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_32f16s_Sfs(const Ipp32f* pSrc, Ipp16s* pDst, int
    len, IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_32f16u_Sfs(const Ipp32f* pSrc, Ipp16u* pDst, int
    len, IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_32f32s_Sfs(const Ipp32f* pSrc, Ipp32s* pDst, int
    len, IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_64s32s_Sfs(const Ipp64s* pSrc, Ipp32s* pDst, int
    len, IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_64f32s_Sfs(const Ipp64f* pSrc, Ipp32s* pDst, int
    len, IppRoundMode rndMode, int scaleFactor);


IppStatus ippsConvert_24u32u(const Ipp8u* pSrc, Ipp32u* pDst, int len);
IppStatus ippsConvert_24u32f(const Ipp8u* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_32u24u_Sfs(const Ipp32u* pSrc, Ipp8u* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32f24u_Sfs(const Ipp32f* pSrc, Ipp8u* pDst, int
    len, int scaleFactor);


IppStatus ippsConvert_24s32s(const Ipp8u* pSrc, Ipp32s* pDst, int len);
IppStatus ippsConvert_24s32f(const Ipp8u* pSrc, Ipp32f* pDst, int len);
IppStatus ippsConvert_32s24s_Sfs(const Ipp32s* pSrc, Ipp8u* pDst, int
    len, int scaleFactor);
IppStatus ippsConvert_32f24s_Sfs(const Ipp32f* pSrc, Ipp8u* pDst, int
    len, int scaleFactor);
```

```
IppStatus ippsConvert_16s16f(const Ipp16s* pSrc, Ipp16f* pDst, int len,
   IppRoundMode rndMode);
IppStatus ippsConvert_32f16f(const Ipp32f* pSrc, Ipp16f* pDst, int len,
   IppRoundMode rndMode);
IppStatus ippsConvert_16f16s_Sfs(const Ipp16f* pSrc, Ipp16s* pDst, int
   len, IppRoundMode rndMode, int scaleFactor);
IppStatus ippsConvert_16f32f(const Ipp16f* pSrc, Ipp32f* pDst, int len);
```

Join

Converts the floating-point data of several vectors to integer data, and stores the results in a single vector.

```
IppStatus ippsJoin_32f16s_D2L(const Ipp32f** pSrc, int nChannels, int
   chanLen, Ipp16s* pDst);
```

JoinScaled

Converts with scaling the floating-point data of several vectors to integer data and stores the results in a single vector.

```
IppStatus ippsJoinScaled_32f16s_D2L(const Ipp32f** pSrc, int nChannels,
   int chanLen, Ipp16s* pDst);
IppStatus ippsJoinScaled_32f24s_D2L(const Ipp32f** pSrc, int nChannels,
   int chanLen, Ipp8u* pDst);
```

SplitScaled

Converts the integer data of a vector to floating-point data with scaling, and stores the result in several vectors.

```
IppStatus ippsSplitScaled_16s32f_D2L(const Ipp16s* pSrc, Ipp32f** pDst,
   int nChannels, int chanLen);
IppStatus ippsSplitScaled_24s32f_D2L(const Ipp8u* pSrc, Ipp32f** pDst,
   int nChannels, int chanLen);
```

Conj

Stores the complex conjugate values of a vector in a second vector or in-place.

```
IppStatus ippsConj_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len);
IppStatus ippsConj_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsConj_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
IppStatus ippsConj_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsConj_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsConj_64fc_I(Ipp64fc* pSrcDst, int len);
```

ConjFlip

Computes the complex conjugate of a vector and stores the result in reverse order.

```
IppStatus ippsConjFlip_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len);
IppStatus ippsConjFlip_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
```

```
IppStatus ippsConjFlip_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
```

Magnitude

Computes the magnitudes of the elements of a complex vector.

```
IppStatus ippsMagnitude_32f(const Ipp32f* pSrcRe, const Ipp32f* pSrcIm,
   Ipp32f* pDst, int len);
IppStatus ippsMagnitude_64f(const Ipp64f* pSrcRe, const Ipp64f* pSrcIm,
   Ipp64f* pDst, int len);
IppStatus ippsMagnitude_32fc(const Ipp32fc* pSrc, Ipp32f* pDst, int len);
IppStatus ippsMagnitude_64fc(const Ipp64fc* pSrc, Ipp64f* pDst, int len);
IppStatus ippsMagnitude_16s32f(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp32f* pDst, int len);
IppStatus ippsMagnitude_16sc32f(const Ipp16sc* pSrc, Ipp32f* pDst, int
   len);
IppStatus ippsMagnitude_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsMagnitude_16sc_Sfs(const Ipp16sc* pSrc, Ipp16s* pDst, int
   len, int scaleFactor);
IppStatus ippsMagnitude_32sc_Sfs(const Ipp32sc* pSrc, Ipp32s* pDst, int
   len, int scaleFactor);
```

MagSquared

Computes the squared magnitudes of the elements of a complex vector.

```
IppStatus ippsMagSquared_32sc32s_Sfs(const Ipp32sc* pSrc, Ipp32s* pDst,
   int len, int scaleFactor);
```

Phase

Computes the phase angles of elements of a complex vector.

```
IppStatus ippsPhase_64fc(const Ipp64fc* pSrc, Ipp64f* pDst, int len);
IppStatus ippsPhase_32fc(const Ipp32fc* pSrc, Ipp32f* pDst, int len);
IppStatus ippsPhase_16sc32f(const Ipp16sc* pSrc, Ipp32f* pDst, int len);
IppStatus ippsPhase_16sc_Sfs(const Ipp16sc* pSrc, Ipp16s* pDst, int len,
   int scaleFactor);
IppStatus ippsPhase_32sc_Sfs(const Ipp32sc* pSrc, Ipp32s* pDst, int len,
   int scaleFactor);
IppStatus ippsPhase_64f(const Ipp64f* pSrcRe, const Ipp64f* pSrcIm,
   Ipp64f* pDst, int len);
IppStatus ippsPhase_32f(const Ipp32f* pSrcRe, const Ipp32f* pSrcIm,
   Ipp32f* pDst, int len);
IppStatus ippsPhase_16s32f(const Ipp16s* pSrcRe, const Ipp16s* pSrcIm,
   Ipp32f* pDst, int len);
IppStatus ippsPhase_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s* pSrcIm,
   Ipp16s* pDst, int len, int scaleFactor);
```

PowerSpectr

Computes the power spectrum of a complex vector.

```
IppStatus ippsPowerSpectr_64fc(const Ipp64fc* pSrc, Ipp64f* pDst, int
   len);
IppStatus ippsPowerSpectr_32fc(const Ipp32fc* pSrc, Ipp32f* pDst, int
   len);
IppStatus ippsPowerSpectr_16sc_Sfs(const Ipp16sc* pSrc, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsPowerSpectr_16sc32f(const Ipp16sc* pSrc, Ipp32f* pDst, int
   len);
IppStatus ippsPowerSpectr_64f(const Ipp64f* pSrcRe, const Ipp64f*
   pSrcIm, Ipp64f* pDst, int len);
IppStatus ippsPowerSpectr_32f(const Ipp32f* pSrcRe, const Ipp32f*
   pSrcIm, Ipp32f* pDst, int len);
IppStatus ippsPowerSpectr_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsPowerSpectr_16s32f(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp32f* pDst, int len);
```

Real

Returns the real part of a complex vector in a second vector.

```
IppStatus ippsReal_16sc(const Ipp16sc* pSrc, Ipp16s* pDstRe, int len);
IppStatus ippsReal_32fc(const Ipp32fc* pSrc, Ipp32f* pDstRe, int len);
IppStatus ippsReal_64fc(const Ipp64fc* pSrc, Ipp64f* pDstRe, int len);
```

Imag

Returns the imaginary part of a complex vector in a second vector.

```
IppStatus ippsImag_16sc(const Ipp16sc* pSrc, Ipp16s* pDstIm, int len);
IppStatus ippsImag_32fc(const Ipp32fc* pSrc, Ipp32f* pDstIm, int len);
IppStatus ippsImag_64fc(const Ipp64fc* pSrc, Ipp64f* pDstIm, int len);
```

RealToCplx

Returns a complex vector constructed from the real and imaginary parts of two real vectors.

```
IppStatus ippsRealToCplx_16s(const Ipp16s* pSrcRe, const Ipp16s* pSrcIm,
   Ipp16sc* pDst, int len);
IppStatus ippsRealToCplx_32f(const Ipp32f* pSrcRe, const Ipp32f* pSrcIm,
   Ipp32fc* pDst, int len);
IppStatus ippsRealToCplx_64f(const Ipp64f* pSrcRe, const Ipp64f* pSrcIm,
   Ipp64fc* pDst, int len);
```

CplxToReal

Returns the real and imaginary parts of a complex vector in two respective vectors.

```
IppStatus ippsCplxToReal_16sc(const Ipp16sc* pSrc, Ipp16s* pDstRe,
   Ipp16s* pDstIm, int len);
IppStatus ippsCplxToReal_32fc(const Ipp32fc* pSrc, Ipp32f* pDstRe,
   Ipp32f* pDstIm, int len);
IppStatus ippsCplxToReal_64fc(const Ipp64fc* pSrc, Ipp64f* pDstRe,
   Ipp64f* pDstIm, int len);
```

Threshold

Performs the threshold operation on the elements of a vector by limiting the element values by `level`.

```
IppStatus ippsThreshold_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   Ipp16s level, IppCmpOp relOp);
IppStatus ippsThreshold_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f level, IppCmpOp relOp);
IppStatus ippsThreshold_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   Ipp64f level, IppCmpOp relOp);
IppStatus ippsThreshold_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   Ipp32f level, IppCmpOp relOp);
IppStatus ippsThreshold_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len,
   Ipp64f level, IppCmpOp relOp);
IppStatus ippsThreshold_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len,
   Ipp16s level, IppCmpOp relOp);
IppStatus ippsThreshold_16s_I(Ipp16s* pSrcDst, int len, Ipp16s level,
   IppCmpOp relOp);
IppStatus ippsThreshold_32f_I(Ipp32f* pSrcDst, int len, Ipp32f level,
   IppCmpOp relOp);
IppStatus ippsThreshold_64f_I(Ipp64f* pSrcDst, int len, Ipp64f level,
   IppCmpOp relOp);
IppStatus ippsThreshold_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f level,
   IppCmpOp relOp);
IppStatus ippsThreshold_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f level,
   IppCmpOp relOp);
IppStatus ippsThreshold_16sc_I(Ipp16sc* pSrcDst, int len, Ipp16s level,
   IppCmpOp relOp);
```

Threshold_LT, Threshold_GT

Performs the threshold operation on the elements of a vector by limiting the element values by `level`.

```
IppStatus ippsThreshold_LT_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   Ipp16s level);
```

```
IppStatus ippsThreshold_LT_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len,
   Ipp32s level);
IppStatus ippsThreshold_LT_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f level);
IppStatus ippsThreshold_LT_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   Ipp64f level);
IppStatus ippsThreshold_LT_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
   len, Ipp32f level);
IppStatus ippsThreshold_LT_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
   len, Ipp64f level);
IppStatus ippsThreshold_LT_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, Ipp16s level);
IppStatus ippsThreshold_LT_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
   level);
IppStatus ippsThreshold_LT_32s_I(Ipp32s* pSrcDst, int len, Ipp32s
   level);
IppStatus ippsThreshold_LT_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
   level);
IppStatus ippsThreshold_LT_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
   level);
IppStatus ippsThreshold_LT_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f
   level);
IppStatus ippsThreshold_LT_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f
   level);
IppStatus ippsThreshold_LT_16sc_I(Ipp16sc* pSrcDst, int len, Ipp16s
   level);
IppStatus ippsThreshold_GT_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   Ipp16s level);
IppStatus ippsThreshold_GT_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len,
   Ipp32s level);
IppStatus ippsThreshold_GT_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f level);
IppStatus ippsThreshold_GT_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   Ipp64f level);
IppStatus ippsThreshold_GT_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
   len, Ipp32f level);
IppStatus ippsThreshold_GT_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
   len, Ipp64f level);
IppStatus ippsThreshold_GT_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, Ipp16s level);
IppStatus ippsThreshold_GT_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
   level);
```

```
IppStatus ippsThreshold_GT_32s_I(Ipp32s* pSrcDst, int len, Ipp32s
    level);
IppStatus ippsThreshold_GT_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_GT_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
    level);
IppStatus ippsThreshold_GT_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_GT_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f
    level);
IppStatus ippsThreshold_GT_16sc_I(Ipp16sc* pSrcDst, int len, Ipp16s
    level);
```

Threshold_LTAbs, Threshold_GTAbs

Performs the threshold operation on the absolute values of elements of a vector.

```
IppStatus ippsThreshold_LTAbs_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
    len, Ipp16s level);
IppStatus ippsThreshold_LTAbs_32s(const Ipp32s* pSrc, Ipp32s* pDst, int
    len, Ipp32s level);
IppStatus ippsThreshold_LTAbs_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    len, Ipp32f level);
IppStatus ippsThreshold_LTAbs_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len, Ipp64f level);
IppStatus ippsThreshold_LTAbs_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
    level);
IppStatus ippsThreshold_LTAbs_32s_I(Ipp32s* pSrcDst, int len, Ipp32s
    level);
IppStatus ippsThreshold_LTAbs_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_LTAbs_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
    level);


IppStatus ippsThreshold_GTAbs_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
    len, Ipp16s level);
IppStatus ippsThreshold_GTAbs_32s(const Ipp32s* pSrc, Ipp32s* pDst, int
    len, Ipp32s level);
IppStatus ippsThreshold_GTAbs_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    len, Ipp32f level);
IppStatus ippsThreshold_GTAbs_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len, Ipp64f level);
IppStatus ippsThreshold_GTAbs_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
    level);
```

```
IppStatus ippsThreshold_GTAbs_32s_I(Ipp32s* pSrcDst, int len, Ipp32s
    level);
IppStatus ippsThreshold_GTAbs_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_GTAbs_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
    level);
```

Threshold_LTVal, Threshold_GTVal, Threshold_LTValGTVal

Performs the threshold operation on the elements of a vector by limiting the element values by `level`.

```
IppStatus ippsThreshold_LTVal_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
    len, Ipp16s level, Ipp16s value);
IppStatus ippsThreshold_LTVal_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    len, Ipp32f level, Ipp32f value);
IppStatus ippsThreshold_LTVal_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len, Ipp64f level, Ipp64f value);
IppStatus ippsThreshold_LTVal_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst,
    int len, Ipp16s level, Ipp16sc value);
IppStatus ippsThreshold_LTVal_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
    int len, Ipp32f level, Ipp32fc value);
IppStatus ippsThreshold_LTVal_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
    int len, Ipp64f level, Ipp64fc value);
IppStatus ippsThreshold_LTVal_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
    level, Ipp16s value);
IppStatus ippsThreshold_LTVal_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
    level, Ipp32f value);
IppStatus ippsThreshold_LTVal_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
    level, Ipp64f value);
IppStatus ippsThreshold_LTVal_16sc_I(Ipp16sc* pSrcDst, int len, Ipp16s
    level, Ipp16sc value);
IppStatus ippsThreshold_LTVal_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f
    level, Ipp32fc value);
IppStatus ippsThreshold_LTVal_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f
    level, Ipp64fc value);
IppStatus ippsThreshold_GTVal_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
    len, Ipp16s level, Ipp16s value);
IppStatus ippsThreshold_GTVal_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    len, Ipp32f level, Ipp32f value);
IppStatus ippsThreshold_GTVal_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len, Ipp64f level, Ipp64f value);
```

```
IppStatus ippsThreshold_GTVal_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst,
   int len, Ipp16s level, Ipp16sc value);
IppStatus ippsThreshold_GTVal_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
   int len, Ipp32f level, Ipp32fc value);
IppStatus ippsThreshold_GTVal_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
   int len, Ipp64f level, Ipp64fc value);
IppStatus ippsThreshold_GTVal_16s_I(Ipp16s* pSrcDst, int len, Ipp16s
   level, Ipp16s value);
IppStatus ippsThreshold_GTVal_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
   level, Ipp32f value);
IppStatus ippsThreshold_GTVal_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
   level, Ipp64f value);
IppStatus ippsThreshold_GTVal_16sc_I(Ipp16sc* pSrcDst, int len, Ipp16s
   level, Ipp16sc value);
IppStatus ippsThreshold_GTVal_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f
   level, Ipp32fc value);
IppStatus ippsThreshold_GTVal_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f
   level, Ipp64fc value);
IppStatus ippsThreshold_LTValGTVal_16s(const Ipp16s* pSrc, Ipp16s* pDst,
   int len, Ipp16s levelLT, Ipp16s valueLT, Ipp16s levelGT, Ipp16s
   valueGT);
IppStatus ippsThreshold_LTValGTVal_32s(const Ipp32s* pSrc, Ipp32s* pDst,
   int len, Ipp32s levelLT, Ipp32s valueLT, Ipp32s levelGT, Ipp32s
   valueGT);
IppStatus ippsThreshold_LTValGTVal_32f(const Ipp32f* pSrc, Ipp32f* pDst,
   int len, Ipp32f levelLT, Ipp32f valueLT, Ipp32f levelGT, Ipp32f
   valueGT);
IppStatus ippsThreshold_LTValGTVal_64f(const Ipp64f* pSrc, Ipp64f* pDst,
   int len, Ipp64f levelLT, Ipp64f valueLT, Ipp64f levelGT, Ipp64f
   valueGT);
IppStatus ippsThreshold_LTValGTVal_16s_I(Ipp16s* pSrcDst, int len,
   Ipp16s levelLT, Ipp16s valueLT, Ipp16s levelGT, Ipp16s valueGT);
IppStatus ippsThreshold_LTValGTVal_32s_I(Ipp32s* pSrcDst, int len,
   Ipp32s levelLT, Ipp32s valueLT, Ipp32s levelGT, Ipp32s valueGT);
IppStatus ippsThreshold_LTValGTVal_32f_I(Ipp32f* pSrcDst, int len,
   Ipp32f levelLT, Ipp32f valueLT, Ipp32f levelGT, Ipp32f valueGT);
IppStatus ippsThreshold_LTValGTVal_64f_I(Ipp64f* pSrcDst, int len,
   Ipp64f levelLT, Ipp64f valueLT, Ipp64f levelGT, Ipp64f valueGT);
```

Threshold_LTInv

Computes the inverse of vector elements after limiting their magnitudes by the given lower bound.

```
IppStatus ippsThreshold_LTInv_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len, Ipp32f level);
```

```
IppStatus ippsThreshold_LTInv_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len, Ipp64f level);
IppStatus ippsThreshold_LTInv_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
    int len, Ipp32f level);
IppStatus ippsThreshold_LTInv_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
    int len, Ipp64f level);
IppStatus ippsThreshold_LTInv_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_LTInv_64f_I(Ipp64f* pSrcDst, int len, Ipp64f
    level);
IppStatus ippsThreshold_LTInv_32fc_I(Ipp32fc* pSrcDst, int len, Ipp32f
    level);
IppStatus ippsThreshold_LTInv_64fc_I(Ipp64fc* pSrcDst, int len, Ipp64f
    level);
```

CartToPolar

Converts the elements of a complex vector to polar coordinate form.

```
IppStatus ippsCartToPolar_32f(const Ipp32f* pSrcRe, const Ipp32f*
    pSrcIm, Ipp32f* pDstMagn, Ipp32f* pDstPhase, int len);
IppStatus ippsCartToPolar_64f(const Ipp64f* pSrcRe, const Ipp64f*
    pSrcIm, Ipp64f* pDstMagn, Ipp64f* pDstPhase, int len);
IppStatus ippsCartToPolar_32fc(const Ipp32fc* pSrc, Ipp32f* pDstMagn,
    Ipp32f* pDstPhase, int len);
IppStatus ippsCartToPolar_64fc(const Ipp64fc* pSrc, Ipp64f* pDstMagn,
    Ipp64f* pDstPhase, int len);
IppStatus ippsCartToPolar_16sc_Sfs(const Ipp16sc* pSrc, Ipp16s*
    pDstMagn, Ipp16s* pDstPhase, int len, int magnScaleFactor, int
    phaseScaleFactor);
```

PolarToCart

Converts the polar form magnitude/phase pairs stored in input vectors to Cartesian coordinate form.

```
IppStatus ippsPolarToCart_32f(const Ipp32f* pSrcMagn, const Ipp32f*
    pSrcPhase, Ipp32f* pDstRe, Ipp32f* pDstIm, int len);
IppStatus ippsPolarToCart_64f(const Ipp64f* pSrcMagn, const Ipp64f*
    pSrcPhase, Ipp64f* pDstRe, Ipp64f* pDstIm, int len);
IppStatus ippsPolarToCart_32fc(const Ipp32f* pSrcMagn, const Ipp32f*
    pSrcPhase, Ipp32fc* pDst, int len);
IppStatus ippsPolarToCart_64fc(const Ipp64f* pSrcMagn, const Ipp64f*
    pSrcPhase, Ipp64fc* pDst, int len);
IppStatus ippsPolarToCart_16sc_Sfs(const Ipp16s* pSrcMagn, const Ipp16s*
    pSrcPhase, Ipp16sc* pDst, int len, int magnScaleFactor, int
    phaseScaleFactor);
```

MaxOrder
Computes the maximum order of a vector.

```
IppStatus ippsMaxOrder_16s(const Ipp16s* pSrc, int len, int* pOrder);
IppStatus ippsMaxOrder_32s(const Ipp32s* pSrc, int len, int* pOrder);
IppStatus ippsMaxOrder_32f(const Ipp32f* pSrc, int len, int* pOrder);
IppStatus ippsMaxOrder_64f(const Ipp64f* pSrc, int len, int* pOrder);
```

Preemphasize
Computes preemphasis of a single precision real signal in-place.

```
IppStatus ippsPreemphasize_16s(Ipp16s* pSrcDst, int len, Ipp32f val);
IppStatus ippsPreemphasize_32f(Ipp32f* pSrcDst, int len, Ipp32f val);
```

Flip
Reverses the order of elements in a vector.

```
IppStatus ippsFlip_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsFlip_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len);
IppStatus ippsFlip_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsFlip_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsFlip_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsFlip_16u_I(Ipp16u* pSrcDst, int len);
IppStatus ippsFlip_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsFlip_64f_I(Ipp64f* pSrcDst, int len);
```

FindNearestOne
Finds an element of the table which is closest to the specified value.

```
IppStatus ippsFindNearestOne_16u(Ipp16u inpVal, Ipp16u* pOutVal, int*
   pOutIndex, const Ipp16u *pTable, int tblLen);
```

FindNearest
Finds table elements that are closest to the elements of the specified vector.

```
IppStatus ippsFindNearest_16u(const Ipp16u* pVals, Ipp16u* pOutVals, int*
   pOutIndexes, int len, const Ipp16u *pTable, int tblLen);
```

Viterbi Decoder Functions

GetVarPointDV
Fills the array with the information about points that are closest to the received point.

```
IppStatus ippsGetVarPointDV_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst,
   Ipp16sc* pVariantPoint, const Ipp8u* pLabel, int state);
```

CalcStatesDV
Calculates the states of the Viterbi decoder.

```
IppStatus ippsCalcStatesDV_16sc(const Ipp16u* pathError, const Ipp8u*
   pNextState, Ipp16u* pBranchError, const Ipp16s* pCurrentSubsetPoint,
   Ipp16s* pPathTable, int state, int presentIndex);
```

BuildSymblTableDV4D
Fills the array with the information
about possible 4D symbols.

```
IppStatus ippsBuildSymblTableDV4D_16sc(const Ipp16sc* pVariantPoint,
   Ipp16sc* pCurrentSubsetPoint, int state, int bitInversion);
```

UpdatePathMetricsDV
Searches for the state with the minimum path metric.

```
IppStatus ippsUpdatePathMetricsDV_16u(Ipp16u* pBranchError, Ipp16u*
   pMinPathError, Ipp8u* pMinSost, Ipp16u* pPathError, int state);
```

Windowing Functions

WinBartlett
Multiplies a vector by a Bartlett windowing function.

```
IppStatus ippsWinBartlett_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsWinBartlett_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsWinBartlett_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsWinBartlett_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len);
IppStatus ippsWinBartlett_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
   len);
IppStatus ippsWinBartlett_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
   len);
IppStatus ippsWinBartlett_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsWinBartlett_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsWinBartlett_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsWinBartlett_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsWinBartlett_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsWinBartlett_64fc_I(Ipp64fc* pSrcDst, int len);
```

WinBlackman
Multiplies a vector by a Blackman windowing function.

```
IppStatus ippsWinBlackmanQ15_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, int alphaQ15);
```

```
IppStatus ippsWinBlackmanQ15_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, int alphaQ15);
IppStatus ippsWinBlackman_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   float alpha);
IppStatus ippsWinBlackman_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len,
   float alpha);
IppStatus ippsWinBlackman_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   float alpha);
IppStatus ippsWinBlackman_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   float alpha);
IppStatus ippsWinBlackman_64f(Ipp64f* pSrc, Ipp64f* pDst, int len, double
   alpha);
IppStatus ippsWinBlackman_64fc(Ipp64fc* pSrc, Ipp64fc* pDst, int len, double
   alpha);
IppStatus ippsWinBlackmanQ15_16s_I(Ipp16s* pSrcDst, int len, int alphaQ15);
IppStatus ippsWinBlackmanQ15_16s_ISfs(Ipp16s* pSrcDst, int len, int alphaQ15,
   int scaleFactor);
IppStatus ippsWinBlackmanQ15_16sc_I(Ipp16sc* pSrcDst, int len, int alphaQ15);
IppStatus ippsWinBlackman_16s_I(Ipp16s* pSrcDst, int len, float alpha);
IppStatus ippsWinBlackman_16sc_I(Ipp16sc* pSrcDst, int len, float alpha);
IppStatus ippsWinBlackman_32f_I(Ipp32f* pSrcDst, int len, float alpha);
IppStatus ippsWinBlackman_32fc_I(Ipp32fc* pSrcDst, int len, float alpha);
IppStatus ippsWinBlackman_64f_I(Ipp64f* pSrcDst, int len, double alpha);
IppStatus ippsWinBlackman_64fc_I(Ipp64fc* pSrcDst, int len, double alpha);
IppStatus ippsWinBlackmanStd_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsWinBlackmanStd_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len);
IppStatus ippsWinBlackmanStd_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsWinBlackmanStd_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
   len);
IppStatus ippsWinBlackmanStd_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsWinBlackmanStd_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
   len);
IppStatus ippsWinBlackmanStd_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsWinBlackmanStd_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsWinBlackmanStd_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsWinBlackmanStd_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsWinBlackmanStd_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsWinBlackmanStd_64fc_I(Ipp64fc* pSrcDst, int len);
```

```
IppStatus ippsWinBlackmanOpt_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
    len);
IppStatus ippsWinBlackmanOpt_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst,
    int len);
IppStatus ippsWinBlackmanOpt_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    len);
IppStatus ippsWinBlackmanOpt_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
    int len);
IppStatus ippsWinBlackmanOpt_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    len);
IppStatus ippsWinBlackmanOpt_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
    int len);
IppStatus ippsWinBlackmanOpt_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsWinBlackmanOpt_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsWinBlackmanOpt_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsWinBlackmanOpt_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsWinBlackmanOpt_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsWinBlackmanOpt_64fc_I(Ipp64fc* pSrcDst, int len);
```

WinHamming

Multiplies a vector by a Hamming windowing function.

```
IppStatus ippsWinHamming_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsWinHamming_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsWinHamming_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsWinHamming_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
    len);
IppStatus ippsWinHamming_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
    len);
IppStatus ippsWinHamming_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
    len);
IppStatus ippsWinHamming_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsWinHamming_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsWinHamming_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsWinHamming_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsWinHamming_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsWinHamming_64fc_I(Ipp64fc* pSrcDst, int len);
```

WinHann

Multiplies a vector by a Hann windowing function.

```
IppStatus ippsWinHann_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len);
IppStatus ippsWinHann_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len);
IppStatus ippsWinHann_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsWinHann_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len);
IppStatus ippsWinHann_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsWinHann_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len);
IppStatus ippsWinHann_16s_I(Ipp16s* pSrcDst, int len);
IppStatus ippsWinHann_16sc_I(Ipp16sc* pSrcDst, int len);
IppStatus ippsWinHann_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsWinHann_32fc_I(Ipp32fc* pSrcDst, int len);
IppStatus ippsWinHann_64f_I(Ipp64f* pSrcDst, int len);
IppStatus ippsWinHann_64fc_I(Ipp64fc* pSrcDst, int len);
```

WinKaiser

Multiplies a vector by a Kaiser windowing function.

```
IppStatus ippsWinKaiser_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   float alpha);
IppStatus ippsWinKaiser_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   float alpha);
IppStatus ippsWinKaiser_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   float alpha);
IppStatus ippsWinKaiser_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int len,
   float alpha);
IppStatus ippsWinKaiser_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   float alpha);
IppStatus ippsWinKaiser_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len,
   float alpha);
IppStatus ippsWinKaiserQ15_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   int alphaQ15);
IppStatus ippsWinKaiserQ15_16sc(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, int alphaQ15);
IppStatus ippsWinKaiser_16s_I(Ipp16s* pSrcDst, int len, float alpha);
IppStatus ippsWinKaiser_32f_I(Ipp32f* pSrcDst, int len, float alpha);
IppStatus ippsWinKaiser_64f_I(Ipp64f* pSrcDst, int len, float alpha);
IppStatus ippsWinKaiser_16sc_I(Ipp16sc* pSrcDst, int len, float alpha);
IppStatus ippsWinKaiser_32fc_I(Ipp32fc* pSrcDst, int len, float alpha);
```

```
IppStatus ippsWinKaiser_64fc_I(Ipp64fc* pSrcDst, int len, float alpha);
IppStatus ippsWinKaiserQ15_16s_I(Ipp16s* pSrcDst, int len, int
   alphaQ15);
IppStatus ippsWinKaiserQ15_16sc_I(Ipp16sc* pSrcDst, int len, int
   alphaQ15);
```

Statistical Functions

Sum

Computes the sum of the elements of a vector.

```
IppStatus ippsSum_32f(const Ipp32f* pSrc, int len, Ipp32f* pSum,
   IppHintAlgorithm hint);
IppStatus ippsSum_32fc(const Ipp32fc* pSrc, int len, Ipp32fc* pSum,
   IppHintAlgorithm hint);
IppStatus ippsSum_64f(const Ipp64f* pSrc, int len, Ipp64f* pSum);
IppStatus ippsSum_64fc(const Ipp64fc* pSrc, int len, Ipp64fc* pSum);
IppStatus ippsSum_16s_Sfs(const Ipp16s* pSrc, int len, Ipp16s* pSum, int
   scaleFactor);
IppStatus ippsSum_32s_Sfs(const Ipp32s* pSrc, int len, Ipp32s* pSum, int
   scaleFactor);
IppStatus ippsSum_16s32s_Sfs(const Ipp16s* pSrc, int len, Ipp32s* pSum,
   int scaleFactor);
IppStatus ippsSum_16sc_Sfs(const Ipp16sc* pSrc, int len, Ipp16sc* pSum,
   int scaleFactor);
IppStatus ippsSum_16sc32sc_Sfs(const Ipp16sc* pSrc, int len, Ipp32sc*
   pSum, int scaleFactor);
```

Max

Returns the maximum value of a vector.

```
IppStatus ippsMax_16s(const Ipp16s* pSrc, int len, Ipp16s* pMax);
IppStatus ippsMax_32s(const Ipp32s* pSrc, int len, Ipp32s* pMax);
IppStatus ippsMax_32f(const Ipp32f* pSrc, int len, Ipp32f* pMax);
IppStatus ippsMax_64f(const Ipp64f* pSrc, int len, Ipp64f* pMax);
```

MaxIndx

Returns the maximum value of a vector and the index of the maximum element.

```
IppStatus ippsMaxIndx_16s(const Ipp16s* pSrc, int len, Ipp16s* pMax, int*
   pIndx);
IppStatus ippsMaxIndx_32s(const Ipp32s* pSrc, int len, Ipp32s* pMax, int*
   pIndx);
IppStatus ippsMaxIndx_32f(const Ipp32f* pSrc, int len, Ipp32f* pMax, int*
   pIndx);
```

```
IppStatus ippsMaxIndx_64f(const Ipp64f* pSrc, int len, Ipp64f* pMax, int*
   pIndx);
```

MaxAbs

Returns the maximum absolute value of a vector.

```
IppStatus ippsMaxAbs_16s(const Ipp16s* pSrc, int len, Ipp16s* pMaxAbs);
IppStatus ippsMaxAbs_32s(const Ipp32s* pSrc, int len, Ipp32s* pMaxAbs);
```

MaxAbsIndx

Returns the maximum absolute value of a vector and the index of the corresponding element.

```
IppStatus ippsMaxAbsIndx_16s(const Ipp16s* pSrc, int len, Ipp16s*
   pMaxAbs, int* pIndx);
IppStatus ippsMaxAbsIndx_32s(const Ipp32s* pSrc, int len, Ipp32s*
   pMaxAbs, int* pIndx);
```

Min

Returns the minimum value of a vector.

```
IppStatus ippsMin_16s(const Ipp16s* pSrc, int len, Ipp16s* pMin);
IppStatus ippsMin_32s(const Ipp32s* pSrc, int len, Ipp32s* pMin);
IppStatus ippsMin_32f(const Ipp32f* pSrc, int len, Ipp32f* pMin);
IppStatus ippsMin_64f(const Ipp64f* pSrc, int len, Ipp64f* pMin);
```

MinIndx

Returns the minimum value of a vector and the index of the minimum element.

```
IppStatus ippsMinIndx_16s(const Ipp16s* pSrc, int len, Ipp16s* pMin, int*
   pIndx);
IppStatus ippsMinIndx_32s(const Ipp32s* pSrc, int len, Ipp32s* pMin, int*
   pIndx);
IppStatus ippsMinIndx_32f(const Ipp32f* pSrc, int len, Ipp32f* pMin, int*
   pIndx);
IppStatus ippsMinIndx_64f(const Ipp64f* pSrc, int len, Ipp64f* pMin, int*
   pIndx);
```

MinAbs

Returns the minimum absolute value of a vector.

```
IppStatus ippsMinAbs_16s(const Ipp16s* pSrc, int len, Ipp16s* pMinAbs);
IppStatus ippsMinAbs_32s(const Ipp32s* pSrc, int len, Ipp32s* pMinAbs);
```

MinAbsIndx

Returns the minimum absolute value of a vector and the index of the corresponding element.

```
IppStatus ippsMinAbsIndx_16s(const Ipp16s* pSrc, int len, Ipp16s*
   pMinAbs, int* pIndx);
IppStatus ippsMinAbsIndx_32s(const Ipp32s* pSrc, int len, Ipp32s*
   pMinAbs, int* pIndx);
```

MinMax

Returns the maximum and minimum values of a vector.

```
IppStatus ippsMinMax_8u(const Ipp8u* pSrc, int len, Ipp8u* pMin, Ipp8u*
   pMax);
IppStatus ippsMinMax_16u(const Ipp16u* pSrc, int len, Ipp16u* pMin,
   Ipp16u* pMax);
IppStatus ippsMinMax_16s(const Ipp16s* pSrc, int len, Ipp16s* pMin,
   Ipp16s* pMax);
IppStatus ippsMinMax_32u(const Ipp32u* pSrc, int len, Ipp32u* pMin,
   Ipp32u* pMax);
IppStatus ippsMinMax_32s(const Ipp32s* pSrc, int len, Ipp32s* pMin,
   Ipp32s* pMax);
IppStatus ippsMinMax_32f(const Ipp32f* pSrc, int len, Ipp32f* pMin,
   Ipp32f* pMax);
IppStatus ippsMinMax_64f(const Ipp64f* pSrc, int len, Ipp64f* pMin,
   Ipp64f* pMax);
```

MinMaxIndx

Returns the maximum and minimum values of a vector and the indexes of the corresponding elements.

```
IppStatus ippsMinMaxIndx_8u(const Ipp8u* pSrc, int len, Ipp8u* pMin, int*
   pMinIndx, Ipp8u* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_16u(const Ipp16u* pSrc, int len, Ipp16u* pMin,
   int* pMinIndx, Ipp16u* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_16s(const Ipp16s* pSrc, int len, Ipp16s* pMin,
   int* pMinIndx, Ipp16s* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_32u(const Ipp32u* pSrc, int len, Ipp32u* pMin,
   int* pMinIndx, Ipp32u* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_32s(const Ipp32s* pSrc, int len, Ipp32s* pMin,
   int* pMinIndx, Ipp32s* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_32f(const Ipp32f* pSrc, int len, Ipp32f* pMin,
   int* pMinIndx, Ipp32f* pMax, int* pMaxIndx);
IppStatus ippsMinMaxIndx_64f(const Ipp64f* pSrc, int len, Ipp64f* pMin,
   int* pMinIndx, Ipp64f* pMax, int* pMaxIndx);
```

Mean

Computes the mean value of a vector.

```
IppStatus ippsMean_32f(const Ipp32f* pSrc, int len, Ipp32f* pMean,
   IppHintAlgorithm hint);
IppStatus ippsMean_32fc(const Ipp32fc* pSrc, int len, Ipp32fc* pMean,
   IppHintAlgorithm hint);
IppStatus ippsMean_64f(const Ipp64f* pSrc, int len, Ipp64f* pMean);
IppStatus ippsMean_64fc(const Ipp64fc* pSrc, int len, Ipp64fc* pMean);
```

```
IppStatus ippsMean_16s_Sfs(const Ipp16s* pSrc, int len, Ipp16s* pMean,
   int scaleFactor);
IppStatus ippsMean_16sc_Sfs(const Ipp16sc* pSrc, int len, Ipp16sc* pMean,
   int scaleFactor);
```

StdDev

Computes the standard deviation value of a vector.

```
IppStatus ippsStdDev_32f(const Ipp32f* pSrc, int len, Ipp32f* pStdDev,
   IppHintAlgorithm hint);
IppStatus ippsStdDev_64f(const Ipp64f* pSrc, int len, Ipp64f* pStdDev);
IppStatus ippsStdDev_16s32s_Sfs(const Ipp16s* pSrc, int len, Ipp32s*
   pStdDev, int scaleFactor);
IppStatus ippsStdDev_16s_Sfs(const Ipp16s* pSrc, int len, Ipp16s*
   pStdDev, int scaleFactor);
```

Norm

Computes the C, L1, L2, or L2Sqr norm of a vector.

```
IppStatus ippsNorm_Inf_32f(const Ipp32f* pSrc, int len, Ipp32f* pNorm);
IppStatus ippsNorm_Inf_64f(const Ipp64f* pSrc, int len, Ipp64f* pNorm);
IppStatus ippsNorm_Inf_16s32f(const Ipp16s* pSrc, int len, Ipp32f*
   pNorm);
IppStatus ippsNorm_Inf_32fc32f(const Ipp32fc* pSrc, int len, Ipp32f*
   pNorm);
IppStatus ippsNorm_Inf_64fc64f(const Ipp64fc* pSrc, int len, Ipp64f*
   pNorm);
IppStatus ippsNorm_Inf_16s32s_Sfs(const Ipp16s* pSrc, int len, Ipp32s*
   pNorm, int scaleFactor);


IppStatus ippsNorm_L1_32f(const Ipp32f* pSrc, int len, Ipp32f* pNorm);
IppStatus ippsNorm_L1_64f(const Ipp64f* pSrc, int len, Ipp64f* pNorm);
IppStatus ippsNorm_L1_16s32f(const Ipp16s* pSrc, int len, Ipp32f* pNorm);
IppStatus ippsNorm_L1_32fc64f(const Ipp32fc* pSrc, int len, Ipp64f*
   pNorm);
IppStatus ippsNorm_L1_64fc64f(const Ipp64fc* pSrc, int len, Ipp64f*
   pNorm);
IppStatus ippsNorm_L1_16s32s_Sfs(const Ipp16s* pSrc, int len, Ipp32s*
   pNorm, int scaleFactor);
IppStatus ippsNorm_L1_16s64s_Sfs(const Ipp16s* pSrc, int len, Ipp64s*
   pNorm, int scaleFactor);


IppStatus ippsNorm_L2_32f(const Ipp32f* pSrc, int len, Ipp32f* pNorm);
IppStatus ippsNorm_L2_64f(const Ipp64f* pSrc, int len, Ipp64f* pNorm);
```

```
IppStatus ippsNorm_L2_16s32f(const Ipp16s* pSrc, int len, Ipp32f* pNorm);
IppStatus ippsNorm_L2_32fc64f(const Ipp32fc* pSrc, int len, Ipp64f*
   pNorm);
IppStatus ippsNorm_L2_64fc64f(const Ipp64fc* pSrc, int len, Ipp64f*
   pNorm);
IppStatus ippsNorm_L2_16s32s_Sfs(const Ipp16s* pSrc, int len, Ipp32s*
   pNorm, int scaleFactor);

IppStatus ippsNorm_L2Sqr_16s64s_Sfs(const Ipp16s* pSrc, int len, Ipp64s*
   pNorm, int scaleFactor);
```

NormDiff

Computes the C, L1, L2, or L2Sqr norm of two vectors' difference.

```
IppStatus ippsNormDiff_Inf_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_Inf_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_Inf_16s32f(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_Inf_32fc32f(const Ipp32fc* pSrc1, const Ipp32fc*
   pSrc2, int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_Inf_64fc64f(const Ipp64fc* pSrc1, const Ipp64fc*
   pSrc2, int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_Inf_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pNorm, int scaleFactor);

IppStatus ippsNormDiff_L1_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_L1_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L1_16s32f(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_L1_32fc64f(const Ipp32fc* pSrc1, const Ipp32fc*
   pSrc2, int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L1_64fc64f(const Ipp64fc* pSrc1, const Ipp64fc*
   pSrc2, int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L1_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pNorm, int scaleFactor);
IppStatus ippsNormDiff_L1_16s64s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp64s* pNorm, int scaleFactor);
```

```
IppStatus ippsNormDiff_L2_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_L2_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L2_16s32f(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32f* pNorm);
IppStatus ippsNormDiff_L2_32fc64f(const Ipp32fc* pSrc1, const Ipp32fc*
   pSrc2, int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L2_64fc64f(const Ipp64fc* pSrc1, const Ipp64fc*
   pSrc2, int len, Ipp64f* pNorm);
IppStatus ippsNormDiff_L2_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pNorm, int scaleFactor);


IppStatus ippsNormDiff_L2Sqr_16s64s_Sfs(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, int len, Ipp64s* pNorm, int scaleFactor);
```

DotProd

Computes the dot product of two vectors.

```
IppStatus ippsDotProd_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2, int
   len, Ipp32f* pDp);
IppStatus ippsDotProd_32fc(const Ipp32fc* pSrc1, const Ipp32fc* pSrc2,
   int len, Ipp32fc* pDp);
IppStatus ippsDotProd_32f32fc(const Ipp32f* pSrc1, const Ipp32fc* pSrc2,
   int len, Ipp32fc* pDp);
IppStatus ippsDotProd_32f64f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   int len, Ipp64f* pDp);
IppStatus ippsDotProd_32fc64fc(const Ipp32fc* pSrc1, const Ipp32fc*
   pSrc2, int len, Ipp64fc* pDp);
IppStatus ippsDotProd_32f32fc64fc(const Ipp32f* pSrc1, const Ipp32fc*
   pSrc2, int len, Ipp64fc* pDp);
IppStatus ippsDotProd_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2, int
   len, Ipp64f* pDp);
IppStatus ippsDotProd_64fc(const Ipp64fc* pSrc1, const Ipp64fc* pSrc2,
   int len, Ipp64fc* pDp);
IppStatus ippsDotProd_64f64fc(const Ipp64f* pSrc1, const Ipp64fc* pSrc2,
   int len, Ipp64fc* pDp);
IppStatus ippsDotProd_16s64s(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   int len, Ipp64s* pDp);
IppStatus ippsDotProd_16sc64sc(const Ipp16sc* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp64sc* pDp);
IppStatus ippsDotProd_16s16sc64sc(const Ipp16s* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp64sc* pDp);
```

```
IppStatus ippsDotProd_16s32f(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   int len, Ipp32f* pDp);
IppStatus ippsDotProd_16sc32fc(const Ipp16sc* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp32fc* pDp);
IppStatus ippsDotProd_16s16sc32fc(const Ipp16s* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp32fc* pDp);
IppStatus ippsDotProd_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   int len, Ipp16s* pDp, int scaleFactor);
IppStatus ippsDotProd_16sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp16sc* pDp, int scaleFactor);
IppStatus ippsDotProd_32s_Sfs(const Ipp32s* pSrc1, const Ipp32s* pSrc2,
   int len, Ipp32s* pDp, int scaleFactor);
IppStatus ippsDotProd_32sc_Sfs(const Ipp32sc* pSrc1, const Ipp32sc*
   pSrc2, int len, Ipp32sc* pDp, int scaleFactor);
IppStatus ippsDotProd_16s32s_Sfs(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pDp, int scaleFactor);
IppStatus ippsDotProd_16s16sc32sc_Sfs(const Ipp16s* pSrc1, const
   Ipp16sc* pSrc2, int len, Ipp32sc* pDp, int scaleFactor);
IppStatus ippsDotProd_16s32s32s_Sfs(const Ipp16s* pSrc1, const Ipp32s*
   pSrc2, int len, Ipp32s* pDp, int scaleFactor);
IppStatus ippsDotProd_16s16sc_Sfs(const Ipp16s* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp16sc* pDp, int scaleFactor);
IppStatus ippsDotProd_16sc32sc_Sfs(const Ipp16sc* pSrc1, const Ipp16sc*
   pSrc2, int len, Ipp32sc* pDp, int scaleFactor);
IppStatus ippsDotProd_32s32sc_Sfs(const Ipp32s* pSrc1, const Ipp32sc*
   pSrc2, int len, Ipp32sc* pDp, int scaleFactor);
```

MaxEvery, MinEvery

Computes maximum or minimum value for each pair of elements of two vectors.

```
IppStatus ippsMaxEvery_16s_I(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   len);
IppStatus ippsMaxEvery_32s_I(const Ipp32s* pSrc, Ipp32s* pSrcDst, int
   len);
IppStatus ippsMaxEvery_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int
   len);
IppStatus ippsMinEvery_16s_I(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   len);
IppStatus ippsMinEvery_32s_I(const Ipp32s* pSrc, Ipp32s* pSrcDst, int
   len);
IppStatus ippsMinEvery_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int
   len);
```

Sampling Functions

SampleUp

Up-samples a signal, conceptually increasing its sampling rate by an integer factor.

```
IppStatus ippsSampleUp_16s (const Ipp16s* pSrc, int srcLen, Ipp16s* pDst,
   int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleUp_32f (const Ipp32f* pSrc, int srcLen, Ipp32f* pDst,
   int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleUp_64f (const Ipp64f* pSrc, int srcLen, Ipp64f* pDst,
   int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleUp_16sc (const Ipp16sc* pSrc, int srcLen, Ipp16sc*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleUp_32fc (const Ipp32fc* pSrc, int srcLen, Ipp32fc*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleUp_64fc (const Ipp64fc* pSrc, int srcLen, Ipp64fc*
   pDst, int* pDstLen, int factor, int* pPhase);
```

SampleDown

Down-samples a signal, conceptually decreasing its sampling rate by an integer factor.

```
IppStatus ippsSampleDown_16s(const Ipp16s* pSrc, int srcLen, Ipp16s*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleDown_32f(const Ipp32f* pSrc, int srcLen, Ipp32f*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleDown_64f(const Ipp64f* pSrc, int srcLen, Ipp64f*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleDown_16sc(const Ipp16sc* pSrc, int srcLen, Ipp16sc*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleDown_32fc(const Ipp32fc* pSrc, int srcLen, Ipp32fc*
   pDst, int* pDstLen, int factor, int* pPhase);
IppStatus ippsSampleDown_64fc(const Ipp64fc* pSrc, int srcLen, Ipp64fc*
   Dst, int* pDstLen, int factor, int* pPhase);
```

Filtering Functions

Convolution and Correlation Functions

Conv

Performs finite, linear convolution of two vectors.

```
IppStatus ippsConv_32f(const Ipp32f* pSrc1, int src1Len, const Ipp32f*
   pSrc2, int src2Len, Ipp32f* pDst);
IppStatus ippsConv_64f(const Ipp64f* pSrc1, int src1Len, const Ipp64f*
   pSrc2, int src2Len, Ipp64f* pDst);
```

```
IppStatus ippsConv_16s_Sfs(const Ipp16s* pSrc1, int src1Len, const
   Ipp16s* pSrc2, int src2Len, Ipp16s* pDst, int scaleFactor);
```

ConvBiased

Computes the specified number of elements of the full finite linear convolution of two vectors.

```
IppStatus ippsConvBiased_32f(const Ipp32f* pSrc1, int src1Len, const
   Ipp32f* pSrc2, int src2Len, Ipp32f* pDst, int dstLen, int bias);
```

ConvCyclic

Performs cyclic convolution of two sequences of the fixed size.

```
IppStatus ippsConvCyclic8x8_32f(const Ipp32f* x, const Ipp32f* h, Ipp32f*
   y);
IppStatus ippsConvCyclic4x4_32f32fc(const Ipp32f* x, const Ipp32fc* h,
   Ipp32fc* y);
IppStatus ippsConvCyclic8x8_16s_Sfs(const Ipp16s* x, const Ipp16s* h,
   Ipp16s* y, int scaleFactor);
```

AutoCorr

Estimates normal, biased, and unbiased auto-correlation of a vector and stores the result in a second vector.

```
IppStatus ippsAutoCorr_32f(const Ipp32f* pSrc, int srcLen, Ipp32f* pDst,
   int dstLen);
IppStatus ippsAutoCorr_NormA_32f(const Ipp32f* pSrc, int srcLen, Ipp32f*
   pDst, int dstLen);
IppStatus ippsAutoCorr_NormB_32f(const Ipp32f* pSrc, int srcLen, Ipp32f*
   pDst, int dstLen);
IppStatus ippsAutoCorr_64f(const Ipp64f* pSrc, int srcLen, Ipp64f* pDst,
   int dstLen);
IppStatus ippsAutoCorr_NormA_64f(const Ipp64f* pSrc, int srcLen, Ipp64f*
   pDst, int dstLen);
IppStatus ippsAutoCorr_NormB_64f(const Ipp64f* pSrc, int srcLen, Ipp64f*
   pDst, int dstLen );
IppStatus ippsAutoCorr_32fc(const Ipp32fc* pSrc, int srcLen, Ipp32fc*
   pDst, int dstLen);
IppStatus ippsAutoCorr_NormA_32fc(const Ipp32fc* pSrc, int srcLen,
   Ipp32fc* pDst, int dstLen);
IppStatus ippsAutoCorr_NormB_32fc(const Ipp32fc* pSrc, int srcLen,
   Ipp32fc* pDst, int dstLen);
IppStatus ippsAutoCorr_64fc(const Ipp64fc* pSrc, int srcLen, Ipp64fc*
   pDst, int dstLen);
IppStatus ippsAutoCorr_NormA_64fc(const Ipp64fc* pSrc, int srcLen,
   Ipp64fc* pDst, int dstLen);
IppStatus ippsAutoCorr_NormB_64fc(const Ipp64fc* pSrc, int srcLen,
   Ipp64fc* pDst, int dstLen);
```

```
IppStatus ippsAutoCorr_16s_Sfs(const Ipp16s* pSrc, int srcLen, Ipp16s*
   pDst, int dstLen, int scaleFactor );
IppStatus ippsAutoCorr_NormA_16s_Sfs( const Ipp16s* pSrc, int srcLen,
   Ipp16s* pDst, int dstLen, int scaleFactor );
IppStatus ippsAutoCorr_NormB_16s_Sfs(const Ipp16s* pSrc, int srcLen,
   Ipp16s* pDst, int dstLen, int scaleFactor);
```

CrossCorr

Estimates the cross-correlation of two vectors.

```
IppStatus ippsCrossCorr_32f(const Ipp32f* pSrc1, int len1, const Ipp32f*
   pSrc2, int len2, Ipp32f* pDst, int dstLen, int lowLag);
IppStatus ippsCrossCorr_64f(const Ipp64f* pSrc1, int len1, const Ipp64f*
   pSrc2, int len2, Ipp64f* pDst, int dstLen, int lowLag);
IppStatus ippsCrossCorr_32fc(const Ipp32fc* pSrc1, int len1, const
   Ipp32fc* pSrc2, int len2, Ipp32fc* pDst, int dstLen, int lowLag);
IppStatus ippsCrossCorr_64fc(const Ipp64fc* pSrc1, int len1, const
   Ipp64fc* pSrc2, int len2, Ipp64fc* pDst, int dstLen, int lowLag);
IppStatus ippsCrossCorr_16s_Sfs(const Ipp16s* pSrc1, int len1, const
   Ipp16s* pSrc2, int len2, Ipp16s* pDst, int dstLen, int lowLag, int
   scaleFactor);
```

UpdateLinear

Integrates an input vector with specified integration weight.

```
IppStatus ippsUpdateLinear_16s32s_I(const Ipp16s* pSrc, int len, Ipp32s*
   pSrcDst, int srcShiftRight, Ipp16s alpha, IppHintAlgorithm hint);
```

UpdatePower

Integrates the square of an input vector with specified integration weight.

```
IppStatus ippsUpdatePower_16s32s_I(const Ipp16s* pSrc, int len, Ipp32s*
   pSrcDst, int srcShiftRight,Ipp16s alpha, IppHintAlgorithm hint);
```

Filtering Functions

SumWindow

Sums elements in the mask applied to each element of a vector.

```
IppStatus ippsSumWindow_8u32f(const Ipp8u* pSrc, Ipp32f* pDst, int len,
   int maskSize);
IppStatus ippsSumWindow_16s32f(const Ipp16s* pSrc, Ipp32f* pDst, int len,
   int maskSize);
```

FIRInitAlloc

Allocates memory and initializes a single-rate FIR filter state.

```
IppStatus ippsFIRInitAlloc32s_16s(IppsFIRState32s_16s** ppState, const
   Ipp32s* pTaps, int tapsLen, int tapsFactor, const Ipp16s* pDlyLine);
```

```
IppStatus ippsFIRInitAlloc32s_16s32f(IppsFIRState32s_16s** ppState,
   const Ipp32f* pTaps, int tapsLen, const Ipp16s* pDlyLine);
IppStatus ippsFIRInitAlloc32f_16s(IppsFIRState32f_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, const Ipp16s* pDlyLine);
IppStatus ippsFIRInitAlloc64f_16s(IppsFIRState64f_16s** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp16s* pDlyLine);
IppStatus ippsFIRInitAlloc_32s(IppsFIRState_32s** ppState, const Ipp32s*
   pTaps, int tapsLen, const Ipp32s* pDlyLine);
IppStatus ippsFIRInitAlloc64f_32s(IppsFIRState64f_32s** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp32s* pDlyLine);
IppStatus ippsFIRInitAlloc32sc_16sc(IppsFIRState32sc_16sc** ppState,
   const Ipp32sc* pTaps, int tapsLen, int tapsFactor, const Ipp16sc*
   pDlyLine);
IppStatus ippsFIRInitAlloc32sc_16sc32fc(IppsFIRState32sc_16sc** ppState,
   const Ipp32fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine);
IppStatus ippsFIRInitAlloc32fc_16sc(IppsFIRState32fc_16sc** ppState,
   const Ipp32fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine);
IppStatus ippsFIRInitAlloc64fc_16sc(IppsFIRState64fc_16sc** ppState,
   const Ipp64fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine);
IppStatus ippsFIRInitAlloc64fc_32sc(IppsFIRState64fc_32sc** ppState,
   const Ipp64fc* pTaps, int tapsLen, const Ipp32sc* pDlyLine);
IppStatus ippsFIRInitAlloc_32f(IppsFIRState_32f** ppState, const Ipp32f*
   pTaps, int tapsLen, const Ipp32f* pDlyLine);
IppStatus ippsFIRInitAlloc64f_32f(IppsFIRState64f_32f** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp32f* pDlyLine);
IppStatus ippsFIRInitAlloc_64f(IppsFIRState_64f** ppState,const Ipp64f*
   pTaps, int tapsLen, const Ipp64f* pDlyLine);
IppStatus ippsFIRInitAlloc_32fc(IppsFIRState_32fc** ppState, const
   Ipp32fc* pTaps, int tapsLen, const Ipp32fc* pDlyLine);
IppStatus ippsFIRInitAlloc64fc_32fc(IppsFIRState64fc_32fc** ppState,
   const Ipp64fc* pTaps, int tapsLen, const Ipp32fc* pDlyLine);
IppStatus ippsFIRInitAlloc_64fc(IppsFIRState_64fc** ppState, const
   Ipp64fc* pTaps, int tapsLen, const Ipp64fc* pDlyLine);
```

FIRMRInitAlloc

Allocates memory and initializes a multi-rate FIR filter state.

```
IppStatus ippsFIRMRInitAlloc32s_16s(IppsFIRState32s_16s** ppState, const
   Ipp32s* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, const Ipp16s* pDlyLine);
IppStatus ippsFIRMRInitAlloc32s_16s32f(IppsFIRState32s_16s** ppState,
   const Ipp32f *pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine);
```

```
IppStatus ippsFIRMRInitAlloc32f_16s(IppsFIRState32f_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine);

IppStatus ippsFIRMRInitAlloc64f_16s(IppsFIRState64f_16s** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine);

IppStatus ippsFIRMRInitAlloc64f_32s(IppsFIRState64f_32s** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32s* pDlyLine);

IppStatus ippsFIRMRInitAlloc32sc_16sc(IppsFIRState32sc_16sc** ppState,
   const Ipp32sc* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, const Ipp16sc* pDlyLine);

IppStatus ippsFIRMRInitAlloc32sc_16sc32fc(IppsFIRState32sc_16sc**
   ppState, const Ipp32fc* pTaps, int tapsLen, int upFactor, int
   upPhase, int downFactor, int downPhase, const Ipp16sc* pDlyLine);

IppStatus ippsFIRMRInitAlloc32fc_16sc(IppsFIRState32fc_16sc** ppState,
   const Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16sc* pDlyLine);

IppStatus ippsFIRMRInitAlloc64fc_16sc(IppsFIRState64fc_16sc**
   ppState,const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase,
   int downFactor, int downPhase, const Ipp16sc* pDlyLine);

IppStatus ippsFIRMRInitAlloc64fc_32sc(IppsFIRState64fc_32sc** ppState,
   const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32sc* pDlyLine);

IppStatus ippsFIRMRInitAlloc_32f(IppsFIRState_32f** ppState, const
   Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32f* pDlyLine);

IppStatus ippsFIRMRInitAlloc64f_32f(IppsFIRState64f_32f** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32f* pDlyLine);

IppStatus ippsFIRMRInitAlloc_64f(IppsFIRState_64f** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp64f* pDlyLine);

IppStatus ippsFIRMRInitAlloc_32fc(IppsFIRState_32fc** ppState, const
   Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32fc* pDlyLine);

IppStatus ippsFIRMRInitAlloc64fc_32fc(IppsFIRState64fc_32fc** ppState,
   const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32fc* pDlyLine);

IppStatus ippsFIRMRInitAlloc_64fc(IppsFIRState_64fc** ppState, const
   Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp64fc* pDlyLine);
```

FIRFree

Closes a FIR filter state.

```
IppStatus ippsFIRFree32s_16s(IppsFIRState32s_16s* pState);
IppStatus ippsFIRFree32f_16s(IppsFIRState32f_16s* pState);
IppStatus ippsFIRFree64f_16s(IppsFIRState64f_16s* pState);
IppStatus ippsFIRFree_32s(IppsFIRState_32s* pState);
IppStatus ippsFIRFree64f_32s(IppsFIRState64f_32s* pState);
IppStatus ippsFIRFree32sc_16sc(IppsFIRState32sc_16sc* pState);
IppStatus ippsFIRFree32fc_16sc(IppsFIRState32fc_16sc* pState);
IppStatus ippsFIRFree64fc_16sc(IppsFIRState64fc_16sc* pState);
IppStatus ippsFIRFree64fc_32sc(IppsFIRState64fc_32sc* pState);
IppStatus ippsFIRFree_32f(IppsFIRState_32f* pState);
IppStatus ippsFIRFree64f_32f(IppsFIRState64f_32f* pState);
IppStatus ippsFIRFree_64f(IppsFIRState_64f* pState);
IppStatus ippsFIRFree_32fc(IppsFIRState_32fc* pState);
IppStatus ippsFIRFree_64fc(IppsFIRState_64fc* pState);
```

FIRInit

Initializes a single-rate FIR filter state.

```
IppStatus ippsFIRInit32s_16s(IppsFIRState32s_16s** ppState, const
   Ipp32s* pTaps, int tapsLen, int tapsFactor, const Ipp16s* pDlyLine,
   Ipp8u* pBuffer);
IppStatus ippsFIRInit32s_16s32f(IppsFIRState32s_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit32f_16s(IppsFIRState32f_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit64f_16s(IppsFIRState64f_16s** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit_32s(IppsFIRState_32s** ppState, const Ipp32s*
   pTaps, int tapsLen, const Ipp32s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit64f_32s(IppsFIRState64f_32s** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp32s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit32sc_16sc(IppsFIRState32sc_16sc** ppState, const
   Ipp32sc* pTaps, int tapsLen, int tapsFactor, const Ipp16sc* pDlyLine,
   Ipp8u* pBuffer);
IppStatus ippsFIRInit32sc_16sc32fc(IppsFIRState32sc_16sc** ppState,
   const Ipp32fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine, Ipp8u*
   pBuffer);
```

```
IppStatus ippsFIRInit32fc_16sc(IppsFIRState32fc_16sc** ppState, const
   Ipp32fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine, Ipp8u*
   pBuffer);
IppStatus ippsFIRInit64fc_16sc(IppsFIRState64fc_16sc** ppState, const
   Ipp64fc* pTaps, int tapsLen, const Ipp16sc* pDlyLine,
   Ipp8u* pBuffer);
IppStatus ippsFIRInit64fc_32sc(IppsFIRState64fc_32sc** ppState, const
   Ipp64fc* pTaps, int tapsLen, const Ipp32sc* pDlyLine, Ipp8u*
   pBuffer);
IppStatus ippsFIRInit_32f(IppsFIRState_32f** ppState, const Ipp32f*
   pTaps, int tapsLen, const Ipp32f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit64f_32f(IppsFIRState64f_32f** ppState, const
   Ipp64f* pTaps, int tapsLen, const Ipp32f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit_64f(IppsFIRState_64f** ppState, const Ipp64f*
   pTaps, int tapsLen, const Ipp64f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit_32fc(IppsFIRState_32fc** ppState, const Ipp32fc*
   pTaps, int tapsLen, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRInit64fc_32fc(IppsFIRState64fc_32fc** ppState, const
   Ipp64fc* pTaps, int tapsLen, const Ipp32fc* pDlyLine, Ipp8u*
   pBuffer);
IppStatus ippsFIRInit_64fc(IppsFIRState_64fc** ppState, const Ipp64fc*
   pTaps, int tapsLen, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);
```

FIR MRInit

Initializes a multi-rate FIR filter state.

```
IppStatus ippsFIRMRInit32s_16s(IppsFIRState32s_16s** ppState, const
   Ipp32s* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, const Ipp16s* pDlyLine,
   Ipp8u* pBuffer);
IppStatus ippsFIRMRInit32s_16s32f(IppsFIRState32s_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit32f_16s(IppsFIRState32f_16s** ppState, const
   Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64f_16s(IppsFIRState64f_16s** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp16s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64f_32s(IppsFIRState64f_32s** ppState, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, const Ipp32s* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit32sc_16sc(IppsFIRState32sc_16sc** ppState, const
   Ipp32sc* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, const Ipp16sc* pDlyLine,
   Ipp8u* pBuffer);
```

```
IppStatus ippsFIRMRInit32sc_16sc32fc(IppsFIRState32sc_16sc** ppState,
    const Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp16sc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit32fc_16sc(IppsFIRState32fc_16sc** ppState, const
    Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp16sc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64fc_16sc(IppsFIRState64fc_16sc** ppState, const
    Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp16sc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64fc_32sc(IppsFIRState64fc_32sc** ppState, const
    Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp32sc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit_32f(IppsFIRState_32f** ppState, const Ipp32f*
    pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
    int downPhase, const Ipp32f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64f_32f(IppsFIRState64f_32f** ppState, const
    Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp32f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit_64f(IppsFIRState_64f** ppState, const Ipp64f*
    pTaps, int tapsLen, int upFactor, int upPhase, int downFactor, int
    downPhase, const Ipp64f* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit_32fc(IppsFIRState_32fc** ppState, const Ipp32fc*
    pTaps, int tapsLen, int upFactor, int upPhase, int downFactor, int
    downPhase, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit64fc_32fc(IppsFIRState64fc_32fc** ppState, const
    Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);
IppStatus ippsFIRMRInit_64fc(IppsFIRState_64fc** ppState, const Ipp64fc*
    pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
    int downPhase, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);
```

FIRGetStateSize, FIRMRGetStateSize

Returns the length of the FIR filter state structure.

```
IppStatus ippsFIRGetStateSize_32s(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize_32f(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize_32fc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize_64f(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize_64fc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize32s_16s(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize32s_16s32f(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize32sc_16sc(int tapsLen, int* pBufferSize);
```

```
IppStatus ippsFIRGetStateSize32sc_16sc32fc(int tapsLen, int*
   pBufferSize);
IppStatus ippsFIRGetStateSize32f_16s(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize32fc_16sc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64f_16s(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64f_32f(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64f_32s(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64fc_16sc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64fc_32sc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRGetStateSize64fc_32fc(int tapsLen, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize_32f(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize_32fc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize_64f(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize_64fc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32s_16s(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32s_16s32f(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32sc_16sc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32sc_16sc32fc(int tapsLen, int upFactor,
   int downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32f_16s(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize32fc_16sc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize64f_16s(int tapsLen, int upFactor,  int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize64f_32s(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize64f_32f(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize64fc_16sc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
IppStatus ippsFIRMRGetStateSize64fc_32sc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
```

```
IppStatus ippsFIRMRGetStateSize64fc_32fc(int tapsLen, int upFactor, int
   downFactor, int* pBufferSize);
```

FIRGetTaps

Retrieves the tap values from the FIR filter state structure.

```
IppStatus ippsFIRGetTaps_32s(const IppsFIRState_32s* pState, Ipp32f*
   pTaps);
IppStatus ippsFIRGetTaps_32f(const IppsFIRState_32f* pState, Ipp32f*
   pTaps);
IppStatus ippsFIRGetTaps_64f(const IppsFIRState_64f* pState, Ipp64f*
   pTaps);
IppStatus ippsFIRGetTaps32f_16s(const IppsFIRState32f_16s* pState,
   Ipp32f* pTaps);
IppStatus ippsFIRGetTaps64f_16s(const IppsFIRState64f_16s* pState,
   Ipp64f* pTaps);
IppStatus ippsFIRGetTaps64f_32s(const IppsFIRState64f_32s* pState,
   Ipp64f* pTaps);
IppStatus ippsFIRGetTaps64f_32f(const IppsFIRState64f_32f* pState,
   Ipp64f* pTaps);
IppStatus ippsFIRGetTaps_32fc(const IppsFIRState_32fc* pState, Ipp32fc*
   pTaps);
IppStatus ippsFIRGetTaps_64fc(const IppsFIRState_64fc* pState, Ipp64fc*
   pTaps);
IppStatus ippsFIRGetTaps32fc_16sc(const IppsFIRState32fc_16sc* pState,
   Ipp32fc* pTaps);
IppStatus ippsFIRGetTaps64fc_16sc(const IppsFIRState64fc_16sc* pState,
   Ipp64fc* pTaps);
IppStatus ippsFIRGetTaps64fc_32sc(const IppsFIRState64fc_32sc* pState,
   Ipp64fc* pTaps);
IppStatus ippsFIRGetTaps64fc_32fc(const IppsFIRState64fc_32fc* pState,
   Ipp64fc* pTaps);
IppStatus ippsFIRGetTaps32s_16s32f(const IppsFIRState32s_16s* pState,
   Ipp32f* pTaps);
IppStatus ippsFIRGetTaps32sc_16sc32fc(const IppsFIRState32sc_16sc*
   pState, Ipp32fc* pTaps);
IppStatus ippsFIRGetTaps32s_16s(const IppsFIRState32s_16s* pState,
   Ipp32s* pTaps, int* pTapsFactor);
IppStatus ippsFIRGetTaps32sc_16sc(const IppsFIRState32sc_16sc* pState,
   Ipp32sc* pTaps, int* pTapsFactor);
```

FIRSetTaps

Sets the tap values in the FIR filter state structure.

```
IppStatus ippsFIRSetTaps_32s(const Ipp32s* pTaps, IppsFIRState_32s*
   pState);
```

```
IppStatus ippsFIRSetTaps_32f(const Ipp32f* pTaps, IppsFIRState_32f*
   pState);
IppStatus ippsFIRSetTaps_64f(const Ipp64f* pTaps, IppsFIRState_64f*
   pState);
IppStatus ippsFIRSetTaps32f_16s(const Ipp32f* pTaps,
   IppsFIRState32f_16s* pState);
IppStatus ippsFIRSetTaps64f_16s(const Ipp64f* pTaps,
   IppsFIRState64f_16s* pState);
IppStatus ippsFIRSetTaps64f_32s(const Ipp64f* pTaps,
   IppsFIRState64f_32s* pState);
IppStatus ippsFIRSetTaps64f_32f(const Ipp64f* pTaps,
   IppsFIRState64f_32f* pState);
IppStatus ippsFIRSetTaps_32fc(const Ipp32fc* pTaps, IppsFIRState_32fc*
   pState);
IppStatus ippsFIRSetTaps_64fc(const Ipp64fc* pTaps, IppsFIRState_64fc*
   pState);
IppStatus ippsFIRSetTaps32fc_16sc(const Ipp32fc* pTaps,
   IppsFIRState32fc_16sc* pState);
IppStatus ippsFIRSetTaps64fc_16sc(const Ipp64fc* pTaps,
   IppsFIRState64fc_16sc* pState);
IppStatus ippsFIRSetTaps64fc_32sc(const Ipp64fc* pTaps,
   IppsFIRState64fc_32sc* pState);
IppStatus ippsFIRSetTaps64fc_32fc(const Ipp64fc* pTaps,
   IppsFIRState64fc_32fc* pState);
IppStatus ippsFIRSetTaps32s_16s32f(const Ipp32f* pTaps,
   IppsFIRState32s_16s* pState);
IppStatus ippsFIRSetTaps32sc_16sc32fc(const Ipp32fc* pTaps,
   IppsFIRState32sc_16sc* pState);
IppStatus ippsFIRSetTaps32s_16s(const Ipp32s* pTaps,
   IppsFIRState32s_16s* pState, int tapsFactor);
IppStatus ippsFIRSetTaps32sc_16sc(const Ipp32sc* pTaps,
   IppsFIRState32sc_16sc* pState, int tapsFactor);
```

FIRGetDlyLine

Retrieves the delay line contents from the FIR filter state structuure.

```
IppStatus ippsFIRGetDlyLine_32f(const IppsFIRState_32f* pState, Ipp32f*
   pDlyLine);
IppStatus ippsFIRGetDlyLine_64f(const IppsFIRState_64f* pState, Ipp64f*
   pDlyLine);
IppStatus ippsFIRGetDlyLine32s_16s(const IppsFIRState32s_16s* pState,
   Ipp16s* pDlyLine);
IppStatus ippsFIRGetDlyLine32f_16s(const IppsFIRState32f_16s* pState,
   Ipp16s* pDlyLine);
```

```
IppStatus ippsFIRGetDlyLine64f_16s(const IppsFIRState64f_16s* pState,
   Ipp16s* pDlyLine);
IppStatus ippsFIRGetDlyLine64f_32s(const IppsFIRState64f_32s* pState,
   Ipp32s* pDlyLine);
IppStatus ippsFIRGetDlyLine64f_32f(const IppsFIRState64f_32f* pState,
   Ipp32f* pDlyLine);


IppStatus ippsFIRGetDlyLine_32fc(const IppsFIRState_32fc* pState,
   Ipp32fc* pDlyLine);
IppStatus ippsFIRGetDlyLine_64fc(const IppsFIRState_64fc* pState,
   Ipp64fc* pDlyLine);
IppStatus ippsFIRGetDlyLine32sc_16sc(const IppsFIRState32sc_16sc*
   pState, Ipp16sc* pDlyLine);
IppStatus ippsFIRGetDlyLine32fc_16sc(const IppsFIRState32fc_16sc*
   pState, Ipp16sc* pDlyLine);
IppStatus ippsFIRGetDlyLine64fc_16sc(const IppsFIRState64fc_16sc*
   pState, Ipp16sc* pDlyLine);
IppStatus ippsFIRGetDlyLine64fc_32sc(const IppsFIRState64fc_32sc*
   pState, Ipp32sc* pDlyLine);
IppStatus ippsFIRGetDlyLine64fc_32fc(const IppsFIRState64fc_32fc*
   pState, Ipp32fc* pDlyLine);
```

FIRSetDlyLine

Sets the delay line contents in the FIR filter state structure.

```
IppStatus ippsFIRSetDlyLine_32f(IppsFIRState_32f* pState, const Ipp32f*
   pDlyLine);
IppStatus ippsFIRSetDlyLine_64f(IppsFIRState_64f* pState, const Ipp64f*
   pDlyLine);
IppStatus ippsFIRSetDlyLine32s_16s(IppsFIRState32s_16s* pState, const
   Ipp16s* pDlyLine);
IppStatus ippsFIRSetDlyLine32f_16s(IppsFIRState32f_16s* pState, const
   Ipp16s* pDlyLine);
IppStatus ippsFIRSetDlyLine64f_16s(IppsFIRState64f_16s* pState, const
   Ipp16s* pDlyLine);
IppStatus ippsFIRSetDlyLine64f_32s(IppsFIRState64f_32s* pState, const
   Ipp32s* pDlyLine);
IppStatus ippsFIRSetDlyLine64f_32f(IppsFIRState64f_32f* pState, const
   Ipp32f* pDlyLine);


IppStatus ippsFIRSetDlyLine_32fc(IppsFIRState_32fc* pState, const
   Ipp32fc* pDlyLine);
IppStatus ippsFIRSetDlyLine_64fc(IppsFIRState_64fc* pState, const
   Ipp64fc* pDlyLine);
```

```
IppStatus ippsFIRSetDlyLine32sc_16sc(IppsFIRState32sc_16sc* pState,
   const Ipp16sc* pDlyLine);
IppStatus ippsFIRSetDlyLine32fc_16sc(IppsFIRState32fc_16sc* pState,
   const Ipp16sc* pDlyLine);
IppStatus ippsFIRSetDlyLine64fc_16sc(IppsFIRState64fc_16sc* pState,
   const Ipp16sc* pDlyLine);
IppStatus ippsFIRSetDlyLine64fc_32sc(IppsFIRState64fc_32sc* pState,
   const Ipp32sc* pDlyLine);
IppStatus ippsFIRSetDlyLine64fc_32fc(IppsFIRState64fc_32fc* pState,
   const Ipp32fc* pDlyLine);
```

FIROne

Filters a single sample through a FIR filter.

```
IppStatus ippsFIROne_32s_Sfs(Ipp32s src, Ipp32s* pDstVal,
   IppsFIRStates_32s* pState, int scaleFactor);
IppStatus ippsFIROne32s_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsFIRState32s_16s* pState, int scaleFactor);
IppStatus ippsFIROne32f_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsFIRState32f_16s* pState, int scaleFactor);
IppStatus ippsFIROne64f_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsFIRState64f_16s* pState, int scaleFactor);
IppStatus ippsFIROne64f_32s_Sfs(Ipp32s src, Ipp32s* pDstVal,
   IppsFIRState64f_32s* pState, int scaleFactor);
IppStatus ippsFIROne32sc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsFIRState32sc_16sc* pState, int scaleFactor);
IppStatus ippsFIROne32fc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsFIRState32fc_16sc* pState, int scaleFactor);
IppStatus ippsFIROne64fc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsFIRState64fc_16sc* pState, int scaleFactor);
IppStatus ippsFIROne64fc_32sc_Sfs(Ipp32sc src, Ipp32sc* pDstVal,
   IppsFIRState64fc_32sc* pState, int scaleFactor);
IppStatus ippsFIROne_32f(Ipp32f src, Ipp32f* pDstVal, IppsFIRState_32f*
   pState);
IppStatus ippsFIROne_64f(Ipp64f src, Ipp64f* pDstVal, IppsFIRState_64f*
   pState);
IppStatus ippsFIROne64f_32f(Ipp32f src, Ipp32f* pDstVal,
   IppsFIRState64f_32f* pState);
IppStatus ippsFIROne_32fc(Ipp32fc src, Ipp32fc* pDstVal,
   IppsFIRState_32fc* pState);
IppStatus ippsFIROne_64fc(Ipp64fc src, Ipp64fc* pDstVal,
   IppsFIRState_64fc* pState);
IppStatus ippsFIROne64fc_32fc(Ipp32fc src, Ipp32fc* pDstVal,
   IppsFIRState64fc_32fc* pState);
```

FIR

Filters a block of samples through a single-rate or multi-rate FIR filter.

```
IppStatus ippsFIR_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int numIters,
   IppsFIRState_32s* pState, int scaleFactor);

IppStatus ippsFIR32s_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   numIters, IppsFIRState32s_16s* pState, int scaleFactor);

IppStatus ippsFIR32f_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   numIters, IppsFIRState32f_16s* pState, int scaleFactor);

IppStatus ippsFIR64f_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   numIters, IppsFIRState64f_16s* pState, int scaleFactor);

IppStatus ippsFIR64f_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int
   numIters, IppsFIRState64f_32s* pState, int scaleFactor);

IppStatus ippsFIR32sc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   numIters, IppsFIRState32sc_16sc* pState, int scaleFactor);

IppStatus ippsFIR32fc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   numIters, IppsFIRState32fc_16sc* pState, int scaleFactor);

IppStatus ippsFIR64fc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   numIters, IppsFIRState64fc_16sc* pState, int scaleFactor);

IppStatus ippsFIR64fc_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc* pDst, int
   numIters, IppsFIRState64fc_32sc* pState, int scaleFactor);

IppStatus ippsFIR_32f(const Ipp32f* pSrc, Ipp32f* pDst, int numIters,
   IppsFIRState_32f* pState);

IppStatus ippsFIR_64f(const Ipp64f* pSrc, Ipp64f* pDst, int numIters,
   IppsFIRState_64f* pState);

IppStatus ippsFIR_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int numIters,
   IppsFIRState_32fc* pState);

IppStatus ippsFIR_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int numIters,
   IppsFIRState_64fc* pState);

IppStatus ippsFIR64f_32f(const Ipp32f* pSrc, Ipp32f* pDst, int numIters,
   IppsFIRState64f_32f* pState);

IppStatus ippsFIR64fc_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
   numIters, IppsFIRState64fc_32fc* pState);

IppStatus ippsFIR_32s_ISfs(Ipp32s* pSrcDst, int numIters,
   IppsFIRState_32s* pState, int scaleFactor);

IppStatus ippsFIR32s_16s_ISfs(Ipp16s* pSrcDst, int numIters,
   IppsFIRState32s_16s* pState, int scaleFactor);

IppStatus ippsFIR32f_16s_ISfs(Ipp16s* pSrcDst, int numIters,
   IppsFIRState32f_16s* pState, int scaleFactor);

IppStatus ippsFIR64f_16s_ISfs(Ipp16s* pSrcDst, int numIters,
   IppsFIRState64f_16s* pState, int scaleFactor);

IppStatus ippsFIR64f_32s_ISfs(Ipp32s* pSrcDst, int numIters,
   IppsFIRState64f_32s* pState, int scaleFactor);
```

```
IppStatus ippsFIR32sc_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
   IppsFIRState32sc_16sc* pState, int scaleFactor);
IppStatus ippsFIR32fc_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
   IppsFIRState32fc_16sc* pState, int scaleFactor);
IppStatus ippsFIR64fc_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
   IppsFIRState64fc_16sc* pState, int scaleFactor);
IppStatus ippsFIR64fc_32sc_ISfs(Ipp32sc* pSrcDst, int numIters,
   IppsFIRState64fc_32sc* pState, int scaleFactor);
IppStatus ippsFIR_32f_I(Ipp32f* pSrcDst, int numIters, IppsFIRState_32f*
   pState);
IppStatus ippsFIR_64f_I(Ipp64f* pSrcDst, int numIters, IppsFIRState_64f*
   pState);
IppStatus ippsFIR64f_32f_I(Ipp32f* pSrcDst, int numIters,
   IppsFIRState64f_32f* pState);
IppStatus ippsFIR_32fc_I(Ipp32fc* pSrcDst, int numIters,
   IppsFIRState_32fc* pState);
IppStatus ippsFIR_64fc_I(Ipp64fc* pSrcDst, int numIters,
   IppsFIRState_64fc* pState);
IppStatus ippsFIR64fc_32fc_I(Ipp32fc* pSrcDst, int numIters,
   IppsFIRState64fc_32fc* pState);
```

FIROne_Direct

Directly filters a single sample through a FIR filter.

```
IppStatus ippsFIROne_Direct_16s_Sfs(Ipp16s src, Ipp16s* pDstVal, const
   Ipp16s* pTapsQ15, int tapsLen, Ipp16s* pDlyLine, int* pDlyLineIndex,
   int scaleFactor);
IppStatus ippsFIROne32f_Direct_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   const Ipp32f* pTaps, int tapsLen, Ipp16s* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
IppStatus ippsFIROne64f_Direct_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   const Ipp64f* pTaps, int tapsLen, Ipp16s* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
IppStatus ippsFIROne64f_Direct_32s_Sfs(Ipp32s src, Ipp32s* pDstVal,
   const Ipp64f* pTaps, int tapsLen, Ipp32s* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
IppStatus ippsFIROne32fc_Direct_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   const Ipp32fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
IppStatus ippsFIROne64fc_Direct_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   const Ipp64fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
IppStatus ippsFIROne64fc_Direct_32sc_Sfs(Ipp32sc src, Ipp32sc* pDstVal,
   const Ipp64fc* pTaps, int tapsLen, Ipp32sc* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);
```

```
IppStatus ippsFIROne32s_Direct_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
    const Ipp32s* pTaps, int tapsLen, int tapsFactor, Ipp16s* pDlyLine,
    int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIROne32sc_Direct_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
    const Ipp32sc* pTaps, int tapsLen, int tapsFactor, Ipp16sc* pDlyLine,
    int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIROne_Direct_32f(Ipp32f src, Ipp32f* pDstVal, const
    Ipp32f* pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_64f(Ipp64f src, Ipp64f* pDstVal, const
    Ipp64f* pTaps, int tapsLen, Ipp64f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_32fc(Ipp32fc src, Ipp32fc* pDstVal, const
    Ipp32fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_64fc(Ipp64fc src, Ipp64fc* pDstVal, const
    Ipp64fc* pTaps, int tapsLen, Ipp64fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne64f_Direct_32f(Ipp32f src, Ipp32f* pDstVal, const
    Ipp64f* pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne64fc_Direct_32fc(Ipp32fc src, Ipp32fc* pDstVal,
    const Ipp64fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIROne_Direct_16s_ISfs(Ipp16s* pSrcDstVal, const Ipp16s*
    pTaps, int tapsLen, Ipp16s* pDlyLine, int* pDlyLineIndex, int
    scaleFactor);

IppStatus ippsFIROne32f_Direct_16s_ISfs(Ipp16s* pSrcDstVal, const
    Ipp32f* pTaps, int tapsLen, Ipp16s* pDlyLine, int* pDlyLineIndex, int
    scaleFactor);

IppStatus ippsFIROne64f_Direct_16s_ISfs(Ipp16s* pSrcDstVal, const
    Ipp64f* pTaps, int tapsLen, Ipp16s* pDlyLine, int* pDlyLineIndex, int
    scaleFactor);

IppStatus ippsFIROne64f_Direct_32s_ISfs(Ipp32s* pSrcDstVal, const
    Ipp64f* pTaps, int tapsLen, Ipp32s* pDlyLine, int* pDlyLineIndex, int
    scaleFactor);

IppStatus ippsFIROne32fc_Direct_16sc_ISfs(Ipp16sc* pSrcDstVal, const
    Ipp32fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int* pDlyLineIndex,
    int scaleFactor);

IppStatus ippsFIROne64fc_Direct_16sc_ISfs(Ipp16sc* pSrcDstVal, const
    Ipp64fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int* pDlyLineIndex,
    int scaleFactor);

IppStatus ippsFIROne64fc_Direct_32sc_ISfs(Ipp32sc* pSrcDstVal, const
    Ipp64fc* pTaps, int tapsLen, Ipp32sc* pDlyLine, int* pDlyLineIndex,
    int scaleFactor);

IppStatus ippsFIROne32s_Direct_16s_ISfs(Ipp16s* pSrcDstVal, const
    Ipp32s* pTaps, int tapsLen, int tapsFactor, Ipp16s* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);
```

```
IppStatus ippsFIROne32sc_Direct_16sc_ISfs(Ipp16sc* pSrcDstVal, const
   Ipp32sc* pTaps, int tapsLen, int tapsFactor, Ipp16sc* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);

IppStatus ippsFIROne_Direct_32f_I(Ipp32f* pSrcDstVal, const Ipp32f*
   pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_64f_I(Ipp64f* pSrcDstVal, const Ipp64f*
   pTaps, int tapsLen, Ipp64f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_32fc_I(Ipp32fc* pSrcDstVal, const Ipp32fc*
   pTaps, int tapsLen, Ipp32fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne_Direct_64fc_I(Ipp64fc* pSrcDstVal, const Ipp64fc*
   pTaps, int tapsLen, Ipp64fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne64f_Direct_32f_I(Ipp32f* pSrcDstVal, const Ipp64f*
   pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIROne64fc_Direct_32fc_I(Ipp32fc* pSrcDstVal, const
   Ipp64fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int* pDlyLineIndex);
```

FIR_Direct

Directly filters a block of samples through a single-rate FIR filter.

```
IppStatus ippsFIR_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   numIters, const Ipp16s* pTapsQ15, int tapsLen, Ipp16s* pDlyLine, int*
   pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR32f_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int numIters, const Ipp32f* pTaps, int tapsLen, Ipp16s* pDlyLine,
   int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64f_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int numIters, const Ipp64f* pTaps, int tapsLen, Ipp16s* pDlyLine,
   int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64f_Direct_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   int numIters, const Ipp64f* pTaps, int tapsLen, Ipp32s* pDlyLine,
   int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR32fc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
   pDst, int numIters, const Ipp32fc* pTaps, int tapsLen, Ipp16sc*
   pDlyLine, int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64fc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
   pDst, int numIters, const Ipp64fc* pTaps, int tapsLen, Ipp16sc*
   pDlyLine, int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64fc_Direct_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc*
   pDst, int numIters, const Ipp64fc* pTaps, int tapsLen, Ipp32sc*
   pDlyLine, int* pDlyLineIndex, int scaleFactor);


IppStatus ippsFIR32s_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int numIters, const Ipp32s* pTaps, int tapsLen, int tapsFactor,
   Ipp16s* pDlyLine, int* pDlyLineIndex, int scaleFactor);
```

```
IppStatus ippsFIR32sc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
    pDst, int numIters, const Ipp32sc* pTaps, int tapsLen, int
    tapsFactor, Ipp16sc* pDlyLine, int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR_Direct_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    numIters, const Ipp32f* pTaps, int tapsLen, Ipp32f* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIR_Direct_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    numIters, const Ipp64f* pTaps, int tapsLen, Ipp64f* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIR_Direct_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
    numIters, const Ipp32fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIR_Direct_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
    numIters, const Ipp64fc* pTaps, int tapsLen, Ipp64fc* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIR64f_Direct_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    numIters, const Ipp64f* pTaps, int tapsLen, Ipp32f* pDlyLine, int*
    pDlyLineIndex);

IppStatus ippsFIR64fc_Direct_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
    int numIters, const Ipp64fc* pTaps, int tapsLen, Ipp32fc* pDlyLine,
    int* pDlyLineIndex);

IppStatus ippsFIR_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters, const
    Ipp16s* pTapsQ15, int tapsLen, Ipp16s* pDlyLine, int* pDlyLineIndex,
    int scaleFactor);

IppStatus ippsFIR32f_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
    const Ipp32f* pTaps, int tapsLen, Ipp16s* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64f_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
    const Ipp64f* pTaps, int tapsLen, Ipp16s* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64f_Direct_32s_ISfs(Ipp32s* pSrcDst, int numIters,
    const Ipp64f* pTaps, int tapsLen, Ipp32s* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR32fc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
    const Ipp32fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64fc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
    const Ipp64fc* pTaps, int tapsLen, Ipp16sc* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR64fc_Direct_32sc_ISfs(Ipp32sc* pSrcDst, int numIters,
    const Ipp64fc* pTaps, int tapsLen, Ipp32sc* pDlyLine, int*
    pDlyLineIndex, int scaleFactor);
```

```
IppStatus ippsFIR32s_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
   const Ipp32s* pTaps, int tapsLen, int tapsFactor, Ipp16s* pDlyLine,
   int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR32sc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
   const Ipp32sc* pTaps, int tapsLen, int tapsFactor, Ipp16sc* pDlyLine,
   int* pDlyLineIndex, int scaleFactor);

IppStatus ippsFIR_Direct_32f_I(Ipp32f* pSrcDst, int numIters, const
   Ipp32f* pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIR_Direct_64f_I(Ipp64f* pSrcDst, int numIters, const
   Ipp64f* pTaps, int tapsLen, Ipp64f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIR_Direct_32fc_I(Ipp32fc* pSrcDst, int numIters, const
   Ipp32fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIR_Direct_64fc_I(Ipp64fc* pSrcDst, int numIters, const
   Ipp64fc* pTaps, int tapsLen, Ipp64fc* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIR64f_Direct_32f_I(Ipp32f* pSrcDst, int numIters, const
   Ipp64f* pTaps, int tapsLen, Ipp32f* pDlyLine, int* pDlyLineIndex);

IppStatus ippsFIR64fc_Direct_32fc_I(Ipp32fc* pSrcDst, int numIters,
   const Ipp64fc* pTaps, int tapsLen, Ipp32fc* pDlyLine, int*
   pDlyLineIndex);
```

FIRMR_Direct

Directly filters a block of samples through a multi-rate FIR filter.

```
IppStatus ippsFIRMR32f_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int numIters, const Ipp32f* pTaps, int tapsLen, int upFactor, int
   upPhase, int downFactor, int downPhase, Ipp16s* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR64f_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int numIters, const Ipp64f* pTaps, int tapsLen, int upFactor, int
   upPhase, int downFactor, int downPhase, Ipp16s* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR64f_Direct_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   int numIters, const Ipp64f* pTaps, int tapsLen, int upFactor, int
   upPhase, int downFactor, int downPhase, Ipp32s* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR32fc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
   pDst, int numIters, const Ipp32fc* pTaps, int tapsLen, int upFactor,
   int upPhase, int downFactor, int downPhase, Ipp16sc* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR64fc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
   pDst, int numIters, const Ipp64fc* pTaps, int tapsLen, int upFactor,
   int upPhase, int downFactor, int downPhase, Ipp16sc* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR64fc_Direct_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc*
   pDst, int numIters, const Ipp64fc* pTaps, int tapsLen, int upFactor,
   int upPhase, int downFactor, int downPhase, Ipp32sc* pDlyLine, int
   scaleFactor);
```

```
IppStatus ippsFIRMR32s_Direct_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
    int numIters, const Ipp32s* pTaps, int tapsLen, int tapsFactor, int
    upFactor, int upPhase, int downFactor, int downPhase, Ipp16s*
    pDlyLine, int scaleFactor);

IppStatus ippsFIRMR32sc_Direct_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc*
    pDst, int numIters, const Ipp32sc* pTaps, int tapsLen, int
    tapsFactor, int upFactor, int upPhase, int downFactor, int downPhase,
    Ipp16sc* pDlyLine, int scaleFactor)

IppStatus ippsFIRMR_Direct_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    numIters, const Ipp32f* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp32f* pDlyLine);

IppStatus ippsFIRMR64f_Direct_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
    numIters, const Ipp64f* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp32f* pDlyLine);

IppStatus ippsFIRMR_Direct_64f(const Ipp64f* pSrc, Ipp64f* pDst, int
    numIters, const Ipp64f* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp64f* pDlyLine);

IppStatus ippsFIRMR_Direct_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int
    numIters, const Ipp32fc* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp32fc* pDlyLine);

IppStatus ippsFIRMR64fc_Direct_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
    int numIters, const Ipp64fc* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp32fc* pDlyLine);

IppStatus ippsFIRMR_Direct_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int
    numIters, const Ipp64fc* pTaps, int tapsLen, int upFactor, int
    upPhase, int downFactor, int downPhase, Ipp64fc* pDlyLine);

IppStatus ippsFIRMR32f_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
    const Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp16s* pDlyLine, int scaleFactor);

IppStatus ippsFIRMR64f_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
    const Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp16s* pDlyLine, int scaleFactor);

IppStatus ippsFIRMR64f_Direct_32s_ISfs(Ipp32s* pSrcDst, int numIters,
    const Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp32s* pDlyLine, int scaleFactor);

IppStatus ippsFIRMR32fc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
    const Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp16sc* pDlyLine, int scaleFactor);

IppStatus ippsFIRMR64fc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
    const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp16sc* pDlyLine, int scaleFactor);

IppStatus ippsFIRMR64fc_Direct_32sc_ISfs(Ipp32sc* pSrcDst, int numIters,
    const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
    downFactor, int downPhase, Ipp32sc* pDlyLine, int scaleFactor);
```

```
IppStatus ippsFIRMR32s_Direct_16s_ISfs(Ipp16s* pSrcDst, int numIters,
   const Ipp32s* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, Ipp16s* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR32sc_Direct_16sc_ISfs(Ipp16sc* pSrcDst, int numIters,
   const Ipp32sc* pTaps, int tapsLen, int tapsFactor, int upFactor, int
   upPhase, int downFactor, int downPhase, Ipp16sc* pDlyLine, int
   scaleFactor);

IppStatus ippsFIRMR_Direct_32f_I(Ipp32f* pSrcDst, int numIters, const
   Ipp32f* pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
   int downPhase, Ipp32f* pDlyLine);

IppStatus ippsFIRMR64f_Direct_32f_I(Ipp32f* pSrcDst, int numIters, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
   int downPhase, Ipp32f* pDlyLine);

IppStatus ippsFIRMR_Direct_64f_I(Ipp64f* pSrcDst, int numIters, const
   Ipp64f* pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
   int downPhase, Ipp64f* pDlyLine);)

IppStatus ippsFIRMR_Direct_32fc_I(Ipp32fc* pSrcDst, int numIters, const
   Ipp32fc* pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
   int downPhase, Ipp32fc* pDlyLine);

IppStatus ippsFIRMR64fc_Direct_32fc_I(Ipp32fc* pSrcDst, int numIters,
   const Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int
   downFactor, int downPhase, Ipp32fc* pDlyLine);

IppStatus ippsFIRMR_Direct_64fc_I(Ipp64fc* pSrcDst, int numIters, const
   Ipp64fc* pTaps, int tapsLen, int upFactor, int upPhase, int downFactor,
   int downPhase, Ipp64fc* pDlyLine);
```

FIRSparseInit

Initializes a sparse FIR filter structure.

```
IppStatus ippsFIRSparseInit_32f(IppsFIRSparseState_32f** ppState,   const
   Ipp32f* pNZTaps, const Ipp32s* pNZTapPos, int nzTapsLen, const Ipp32f*
   pDlyLine, Ipp8u* pBuffer);
```

FIRSparseGetStateSize

Computes the size of the external buffer for the sparse FIR filter structure.

```
IppStatus ippsFIRSparseGetStateSize_32f(int nzTapsLen, int order, int*
   pStateSize);
```

FIRSparse

Filters a block of samples through a sparse FIR filter.

```
IppStatus ippsFIRSparse_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   IppsFIRSparseState_32f* pState);
```

FIRGenLowpass
Computes lowpass FIR filter coefficients.

```
IppStatus ippsFIRGenLowpass_64f(Ipp64f rFreq, Ipp64f* taps, int tapsLen,
   IppWinType winType, IppBool doNormal);
```

FIRGenHighpass
Computes highpass FIR filter coefficients.

```
IppStatus ippsFIRGenHighpass_64f(Ipp64f rFreq, Ipp64f* taps, int
   tapsLen, IppWinType winType, IppBool doNormal);
```

FIRGenBandpass
Computes bandpass FIR filter coefficients.

```
IppStatus ippsFIRGenBandpass_64f(Ipp64f rLowFreq, Ipp64f rHighFreq,
   Ipp64f* pTaps, int tapsLen, IppWinType winType, IppBool doNormal);
```

FIRGenBandstop
Computes bandstop FIR filter coefficients.

```
IppStatus ippsFIRGenBandstop_64f(Ipp64f rLowFreq, Ipp64f rHighFreq,
   Ipp64f* pTaps, int tapsLen, IppWinType winType, IppBool doNormal);
```

FIRLMSInitAlloc
Allocates memory and initializes an adaptive FIR filter that uses the least mean squares (LMS) algorithm.

```
IppStatus ippsFIRLMSInitAlloc_32f(IppsFIRLMSState_32f** ppState, const
   Ipp32f* pTaps, int tapsLen, const Ipp32f* pDlyLine, int
   dlyLineIndex);

IppStatus ippsFIRLMSInitAlloc32f_16s(IppsFIRLMSState32f_16s** ppState,
   const Ipp32f* pTaps, int tapsLen, const Ipp16s* pDlyLine, int
   dlyLineIndex);
```

FIRLMSFree
Closes an adaptive FIR filter that uses the least mean squares (LMS) algorithm.

```
IppStatus ippsFIRLMSFree_32f(IppsFIRLMSState_32f* pState);

IppStatus ippsFIRLMSFree32f_16s(IppsFIRLMSState32f_16s* pState);
```

FIRLMSGetTaps
Retrieves the tap values from the FIR LMS filter.

```
IppStatus ippsFIRLMSGetTaps_32f(const IppsFIRLMSState_32f* pState,
   Ipp32f* pOutTaps);

IppStatus ippsFIRLMSGetTaps32f_16s(const IppsFIRLMSState32f_16s* pState,
   Ipp32f* pOutTaps);
```

FIRLMSGetDlyLine
Retrieves the delay line contents from the FIR LMS filter.

```
IppStatus ippsFIRLMSGetDlyLine_32f(const IppsFIRLMSState_32f* pState,
   Ipp32f* pDlyLine, int* pDlyLineIndex);
```

```
IppStatus ippsFIRLMSGetDlyLine32f_16s(const IppsFIRLMSState32f_16s*
   pState, Ipp16s* pDlyLine, int* pDlyLineIndex);
```

FIRLMSSetDlyLine

Sets the delay line contents in the FIR LMS filter.

```
IppStatus ippsFIRLMSGetDlyLine_32f(const IppsFIRLMSState_32f* pState,
   Ipp32f* pDlyLine, int* pDlyLineIndex);
IppStatus ippsFIRLMSGetDlyLine32f_16s(const IppsFIRLMSState32f_16s*
   pState, Ipp16s* pDlyLine, int* pDlyLineIndex);
IppStatus ippsFIRLMSSetDlyLine_32f(IppsFIRLMSState_32f* pState, const
   Ipp32f* pDlyLine, int dlyLineIndex);
IppStatus ippsFIRLMSSetDlyLine32f_16s(IppsFIRLMSState32f_16s* pState,
   const Ipp16s* pDlyLine, int dlyLineIndex);
```

FIRLMS

Filters an array through a FIR LMS filter.

```
IppStatus ippsFIRLMS_32f(const Ipp32f* pSrc, const Ipp32f* pRef, Ipp32f*
   pDst, int len, float mu, IppsFIRLMSState_32f* pState);
IppStatus ippsFIRLMS32f_16s(const Ipp16s* pSrc, const Ipp16s* pRef,
   Ipp16s* pDst, int len, float mu, IppsFIRLMSState32f_16s* pState);
```

FIRLMSOne_Direct

Filters a single sample through a FIR LMS filter.

```
IppStatus ippsFIRLMSOne_Direct_32f(Ipp32f src, Ipp32f refVal, Ipp32f*
   pDstVal, Ipp32f* pTapsInv, int tapsLen, float mu, Ipp32f* pDlyLine,
   int* pDlyLineIndex);
IppStatus ippsFIRLMSOne_Direct32f_16s(Ipp16s src, Ipp16s refVal, Ipp16s*
   pDstVal, Ipp32f* pTapsInv, int tapsLen, float mu, Ipp16s* pDlyLine,
   int* pDlyLineIndex);
IppStatus ippsFIRLMSOne_DirectQ15_16s(Ipp16s src, Ipp16s refVal, Ipp16s*
   pDstVal, Ipp32s* pTapsInv, int tapsLen, int muQ15, Ipp16s* pDlyLine,
   int* pDlyLineIndex);
```

FIRLMSMRInitAlloc

Allocates memory and initializes an adaptive multi-rate FIR filter that uses the least mean squares (LMS) algorithm.

```
IppStatus ippsFIRLMSMRInitAlloc32s_16s(IppsFIRLMSMRState32s_16s**
   ppState, const Ipp32s* pTaps, int tapsLen, const Ipp16s* pDlyLine,
   int dlyLineIndex, int dlyStep, int updateDly, int mu);
IppStatus ippsFIRLMSMRInitAlloc32sc_16sc(IppsFIRLMSMRState32sc_16sc**
   ppState, const Ipp32sc* pTaps, int tapsLen, const Ipp16sc* pDlyLine,
   int dlyLineIndex, int dlyStep, int updateDly, int mu);
```

FIRLMSMRFree

Closes an adaptive multi-rate FIR filter that uses the least mean squares algorithm.

```
IppStatus ippsFIRLMSMRFree32s_16s(IppsFIRLMSMRState32s_16s* pState);
```

```
IppStatus ippsFIRLMSMRFree32sc_16sc(IppsFIRLMSMRState32sc_16sc* pState);
```

FIRLMSMRSetMu

Sets the adaptation step.

```
IppStatus ippsFIRLMSMRSetMu32s_16s(IppsFIRLMSMRState32s_16s* pState,
   const int mu);
IppStatus ippsFIRLMSMRSetMu32sc_16sc(IppsFIRLMSMRState32sc_16sc* pState,
   const int mu);
```

FIRLMSMRUpdateTaps

Updates the filter coefficients using the adaptation error value.

```
IppStatus ippsFIRLMSMRUpdateTaps32s_16s(Ipp32s errVal,
   IppsFIRLMSMRState32s_16s* pState);
IppStatus ippsFIRLMSMRUpdateTaps32sc_16sc(Ipp32sc errVal,
   IppsFIRLMSMRState32sc_16sc* pState);
```

FIRLMSMRGetTaps

Retrieves tap values from the multi-rate FIR LMS filter.

```
IppStatus ippsFIRLMSMRGetTaps32s_16s(IppsFIRLMSMRState32s_16s* pState,
   Ipp32s* pOutTaps);
IppStatus ippsFIRLMSMRGetTaps32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, Ipp32sc* pOutTaps);
```

FIRLMSMRSetTaps

Sets tap values in the multi-rate FIR LMS filter.

```
IppStatus ippsFIRLMSMRSetTaps32s_16s(IppsFIRLMSMRState32s_16s* pState,
   const Ipp32s* pInTaps);
IppStatus ippsFIRLMSMRSetTaps32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, const Ipp32sc* pInTaps);
```

FIRLMSMRGetTapsPointer

Returns the pointer to the filter coefficients.

```
IppStatus ippsFIRLMSMRGetTapsPointer32s_16s(IppsFIRLMSMRState32s_16s*
   pState, Ipp32s** pTaps);
IppStatus
   ippsFIRLMSMRGetTapsPointer32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, Ipp32sc** pTaps);
```

FIRLMSMRGetDlyLine

Retrieves the delay line contents from the multi-rate FIR LMS filter state.

```
IppStatus ippsFIRLMSMRGetDlyLine32s_16s(IppsFIRLMSMRState32s_16s*
   pState, Ipp16s* pOutDlyLine, int* pOutDlyLineIndex);
IppStatus ippsFIRLMSMRGetDlyLine32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, Ipp16sc* pOutDlyLine, int* pOutDlyLineIndex);
```

FIRLMSMRSetDlyLine

Gets and sets the delay line contents of a multi-rate FIR LMS filter state.

```
IppStatus ippsFIRLMSMRSetDlyLine32s_16s(IppsFIRLMSMRState32s_16s*
   pState, const Ipp16s* pInDlyLine, int dlyLineIndex);
IppStatus ippsFIRLMSMRSetDlyLine32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, const Ipp16sc* pInDlyLine, int dlyLineIndex);
```

FIRLMSMRGetDlyVal

Gets one delay line values from the specified position.

```
IppStatus ippsFIRLMSMRGetDlyVal32s_16s(IppsFIRLMSMRState32s_16s* pState,
   Ipp16s* pOutVal, int index);
IppStatus ippsFIRLMSMRGetDlyVal32sc_16sc(IppsFIRLMSMRState32sc_16sc*
   pState, Ipp16sc* pOutVal, int index);
```

FIRLMSMRPutVal

Places the input value in the delay line.

```
IppStatus ippsFIRLMSMRPutVal32s_16s(Ipp16s val,
   IppsFIRLMSMRState32s_16s* pState);
IppStatus ippsFIRLMSMRPutVal32sc_16sc(Ipp16sc val,
   IppsFIRLMSMRState32sc_16sc* pState);
```

FIRLMSMROne

Filters data placed in the delay line.

```
IppStatus ippsFIRLMSMROne32s_16s(Ipp32s* pDstVal,
   IppsFIRLMSMRState32s_16s* pState);
IppStatus ippsFIRLMSMROne32sc_16sc(Ipp32sc* pDstVal,
   IppsFIRLMSMRState32sc_16sc* pState);
```

FIRLMSMROneVal

Filters one input value.

```
IppStatus ippsFIRLMSMROneVal32s_16s(Ipp16s val, Ipp32s* pDstVal,
   IppsFIRLMSMRState32s_16s* pState);
IppStatus ippsFIRLMSMROneVal32sc_16sc(Ipp16sc val, Ipp32sc* pDstVal,
   IppsFIRLMSMRState32sc_16sc* pState);
```

IIRInitAlloc

Allocates memory and initializes the state structure for an arbitraryIIR filter.

```
IppStatus ippsIIRInitAlloc32s_16s(IppsIIRState32s_16s** ppState, const
   Ipp32s* pTaps, int order, int tapsFactor, const Ipp32s* pDlyLine);
IppStatus ippsIIRInitAlloc32s_16s32f(IppsIIRState32s_16s** ppState,
   const Ipp32f* pTaps, int order, const Ipp32s* pDlyLine);
IppStatus ippsIIRInitAlloc32f_16s(IppsIIRState32f_16s** ppState, const
   Ipp32f* pTaps, int order, const Ipp32f* pDlyLine);
IppStatus ippsIIRInitAlloc64f_16s(IppsIIRState64f_16s** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine);
```

```
IppStatus ippsIIRInitAlloc64f_32s(IppsIIRState64f_32s** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine);


IppStatus ippsIIRInitAlloc32sc_16sc(IppsIIRState32sc_16sc** ppState,
   const Ipp32sc* pTaps, int order, int tapsFactor, const Ipp32sc*
   pDlyLine);
IppStatus ippsIIRInitAlloc32sc_16sc32fc(IppsIIRState32sc_16sc** ppState,
   const Ipp32fc* pTaps, int order, const Ipp32sc* pDlyLine);
IppStatus ippsIIRInitAlloc32fc_16sc(IppsIIRState32fc_16sc** ppState,
   const Ipp32fc* pTaps, int order, const Ipp32fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_16sc(IppsIIRState64fc_16sc** ppState,
   const Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_32sc(IppsIIRState64fc_32sc** ppState,
   const Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc_32f(IppsIIRState_32f** ppState, const Ipp32f*
   pTaps, int order, const Ipp32f* pDlyLine);
IppStatus ippsIIRInitAlloc64f_32f(IppsIIRState64f_32f** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine);
IppStatus ippsIIRInitAlloc_64f(IppsIIRState_64f** ppState, const Ipp64f*
   pTaps, int order, const Ipp64f* pDlyLine);
IppStatus ippsIIRInitAlloc_32fc(IppsIIRState_32fc** ppState, const
   Ipp32fc* pTaps, int order, const Ipp32fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_32fc(IppsIIRState64fc_32fc** ppState,
   const Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc_64fc(IppsIIRState_64fc** ppState, const
   Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine);
```

IIRInitAlloc_BiQuad

Allocates memory and initializes the state structure for the biquad IIR filter.

```
IppStatus ippsIIRInitAlloc32s_BiQuad_16s(IppsIIRState32s_16s** ppState,
   const Ipp32s* pTaps, int numBq, int tapsFactor, const Ipp32s*
   pDlyLine);
IppStatus ippsIIRInitAlloc32s_BiQuad_16s32f(IppsIIRState32s_16s**
   ppState, const Ipp32f* pTaps, int numBq, const Ipp32s* pDlyLine);
IppStatus ippsIIRInitAlloc32f_BiQuad_16s(IppsIIRState32f_16s** ppState,
   const Ipp32f* pTaps, int numBq, const Ipp32f* pDlyLine);
IppStatus ippsIIRInitAlloc64f_BiQuad_16s(IppsIIRState64f_16s** ppState,
   const Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine);
IppStatus ippsIIRInitAlloc64f_BiQuad_32s(IppsIIRState64f_32s** ppState,
   const Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine);
```

```
IppStatus ippsIIRInitAlloc32sc_BiQuad_16sc(IppsIIRState32sc_16sc**
   ppState, const Ipp32sc* pTaps, int numBq, int tapsFactor, const
   Ipp32sc* pDlyLine);
IppStatus ippsIIRInitAlloc32sc_BiQuad_16sc32fc(IppsIIRState32sc_16sc**
   ppState, const Ipp32fc* pTaps, int numBq, const Ipp32sc* pDlyLine);
IppStatus ippsIIRInitAlloc32fc_BiQuad_16sc(IppsIIRState32fc_16sc**
   ppState, const Ipp32fc* pTaps, int numBq, const Ipp32fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_BiQuad_16sc(IppsIIRState64fc_16sc**
   ppState, const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_BiQuad_32sc(IppsIIRState64fc_32sc**
   ppState, const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc_BiQuad_32f(IppsIIRState_32f** ppState, const
   Ipp32f* pTaps, int numBq, const Ipp32f* pDlyLine);
IppStatus ippsIIRInitAlloc64f_BiQuad_32f(IppsIIRState64f_32f** ppState,
   const Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine);
IppStatus ippsIIRInitAlloc_BiQuad_64f(IppsIIRState_64f** ppState, const
   Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine);


IppStatus ippsIIRInitAlloc_BiQuad_32fc(IppsIIRState_32fc** ppState,
   const Ipp32fc* pTaps, int numBq, const Ipp32fc* pDlyLine);
IppStatus ippsIIRInitAlloc64fc_BiQuad_32fc(IppsIIRState64fc_32fc**
   ppState, const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine);
IppStatus ippsIIRInitAlloc_BiQuad_64fc(IppsIIRState_64fc** ppState,
   const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine);
```

IIRFree

Closes an IIR filter state.

```
IppStatus ippsIIRFree_32f(IppsIIRState_32f* pState);
IppStatus ippsIIRFree_64f(IppsIIRState_64f* pState);
IppStatus ippsIIRFree_32fc(IppsIIRState_32fc* pState);
IppStatus ippsIIRFree_64fc(IppsIIRState_64fc* pState);
IppStatus ippsIIRFree32s_16s(IppsIIRState32s_16s* pState);
IppStatus ippsIIRFree32sc_16sc(IppsIIRState32sc_16sc* pState);
IppStatus ippsIIRFree32f_16s(IppsIIRState32f_16s* pState);
IppStatus ippsIIRFree32fc_16sc(IppsIIRState32fc_16sc* pState);
IppStatus ippsIIRFree64f_16s(IppsIIRState64f_16s* pState);
IppStatus ippsIIRFree64f_32s(IppsIIRState64f_32s* pState);
IppStatus ippsIIRFree64f_32f(IppsIIRState64f_32f* pState);
IppStatus ippsIIRFree64fc_16sc(IppsIIRState64fc_16sc* pState);
```

```
IppStatus ippsIIRFree64fc_32sc(IppsIIRState64fc_32sc* pState);

IppStatus ippsIIRFree64fc_32fc(IppsIIRState64fc_32fc* pState);
```

IIRInit

Initializes an arbitrary IIR filter state.

```
IppStatus ippsIIRInit32s_16s(IppsIIRState32s_16s** ppState, const
   Ipp32s* pTaps, int order, int tapsFactor, const Ipp32s* pDlyLine,
   Ipp8u* pBuffer);

IppStatus ippsIIRInit32s_16s32f(IppsIIRState32s_16s** ppState, const
   Ipp32f* pTaps, int order, const Ipp32s* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit32f_16s(IppsIIRState32f_16s** ppState, const
   Ipp32f* pTaps, int order, const Ipp32f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_16s(IppsIIRState64f_16s** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_32s(IppsIIRState64f_32s** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine, Ipp8u* pBuffer);


IppStatus ippsIIRInit32sc_16sc(IppsIIRState32sc_16sc** ppState, const
   Ipp32sc* pTaps, int order, int tapsFactor, const Ipp32sc* pDlyLine,
   Ipp8u* pBuffer);

IppStatus ippsIIRInit32sc_16sc32fc(IppsIIRState32sc_16sc** ppState,
   const Ipp32fc* pTaps, int order, const Ipp32sc* pDlyLine, Ipp8u*
   pBuffer);

IppStatus ippsIIRInit32fc_16sc(IppsIIRState32fc_16sc** ppState, const
   Ipp32fc* pTaps, int order, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64fc_16sc(IppsIIRState64fc_16sc** ppState, const
   Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64fc_32sc(IppsIIRState64fc_32sc** ppState, const
   Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit_32f(IppsIIRState_32f** ppState, const Ipp32f*
   pTaps, int order, const Ipp32f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_32f(IppsIIRState64f_32f** ppState, const
   Ipp64f* pTaps, int order, const Ipp64f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit_64f(IppsIIRState_64f** ppState, const Ipp64f*
   pTaps, int order, const Ipp64f* pDlyLine, Ipp8u* pBuffer);


IppStatus ippsIIRInit_32fc(IppsIIRState_32fc** ppState, const Ipp32fc*
   pTaps, int order, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64fc_32fc(IppsIIRState64fc_32fc** ppState, const
   Ipp64fc* pTaps, int order, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit_64fc(IppsIIRState_64fc** ppState, const Ipp64fc*
   pTaps, int order, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);
```

IIRInit_BiQuad

Initializes an IIR filter state.

```
IppStatus ippsIIRInit32s_BiQuad_16s(IppsIIRState32s_16s** ppState, const
   Ipp32s* pTaps, int numBq, int tapsFactor, const Ipp32s* pDlyLine,
   Ipp8u* pBuffer);

IppStatus ippsIIRInit32s_BiQuad_16s32f(IppsIIRState32s_16s** ppState,
   const Ipp32f* pTaps, int numBq, const Ipp32s* pDlyLine, Ipp8u*
   pBuffer);

IppStatus ippsIIRInit32f_BiQuad_16s(IppsIIRState32f_16s** ppState, const
   Ipp32f* pTaps, int numBq, const Ipp32f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_BiQuad_16s(IppsIIRState64f_16s** ppState, const
   Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_BiQuad_32s(IppsIIRState64f_32s** ppState, const
   Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine, Ipp8u* pBuffer);


IppStatus ippsIIRInit32sc_BiQuad_16sc(IppsIIRState32sc_16sc** ppState,
   const Ipp32sc* pTaps, int numBq, int tapsFactor, const Ipp32sc*
   pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit32sc_BiQuad_16sc32fc(IppsIIRState32sc_16sc**
   ppState, const Ipp32fc* pTaps, int numBq, const Ipp32sc* pDlyLine,
   Ipp8u* pBuffer);

IppStatus ippsIIRInit32fc_BiQuad_16sc(IppsIIRState32fc_16sc** ppState,
   const Ipp32fc* pTaps, int numBq, const Ipp32fc* pDlyLine, Ipp8u*
   pBuffer);

IppStatus ippsIIRInit64fc_BiQuad_16sc(IppsIIRState64fc_16sc** ppState,
   const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine, Ipp8u*
   pBuffer);

IppStatus ippsIIRInit64fc_BiQuad_32sc(IppsIIRState64fc_32sc** ppState,
   const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine, Ipp8u*
   pBuffer);

IppStatus ippsIIRInit_BiQuad_32f(IppsIIRState_32f** ppState, const
   Ipp32f* pTaps, int numBq, const Ipp32f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64f_BiQuad_32f(IppsIIRState64f_32f** ppState, const
   Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit_BiQuad_64f(IppsIIRState_64f** ppState, const
   Ipp64f* pTaps, int numBq, const Ipp64f* pDlyLine, Ipp8u* pBuffer);


IppStatus ippsIIRInit_BiQuad_32fc(IppsIIRState_32fc** ppState, const
   Ipp32fc* pTaps, int numBq, const Ipp32fc* pDlyLine, Ipp8u* pBuffer);

IppStatus ippsIIRInit64fc_BiQuad_32fc(IppsIIRState64fc_32fc** ppState,
   const Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine, Ipp8u*
   pBuffer);
```

```
IppStatus ippsIIRInit_BiQuad_64fc(IppsIIRState_64fc** ppState, const
   Ipp64fc* pTaps, int numBq, const Ipp64fc* pDlyLine, Ipp8u* pBuffer);
```

IIRGetStateSize

Computes the length of the external buffer for the arbitrary IIR filter state structure.

```
IppStatus ippsIIRGetStateSize32s_16s(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize32s_16s32f(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize32f_16s(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64f_16s(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64f_32s(int order, int* pBufferSize);

IppStatus ippsIIRGetStateSize32sc_16sc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize32sc_16sc32fc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize32fc_16sc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64fc_16sc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64fc_32sc(int order, int* pBufferSize);

IppStatus ippsIIRGetStateSize_32f(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64f_32f(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize_64f(int order, int* pBufferSize);

IppStatus ippsIIRGetStateSize_32fc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize64fc_32fc(int order, int* pBufferSize);
IppStatus ippsIIRGetStateSize_64fc(int order, int* pBufferSize);
```

IIRGetStateSize_BiQuad

Computes the length of the external buffer for the biquad IIR filter state structure.

```
IppStatus ippsIIRGetStateSize32s_BiQuad_16s(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize32s_BiQuad_16s32f(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize32f_BiQuad_16s(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize64f_BiQuad_16s(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize64f_BiQuad_32s(int numBq, int*
   pBufferSize);

IppStatus ippsIIRGetStateSize32sc_BiQuad_16sc(int numBq, int*
   pBufferSize);
```

```
IppStatus ippsIIRGetStateSize32sc_BiQuad_16sc32fc(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize32fc_BiQuad_16sc(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize64fc_BiQuad_16sc(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize64fc_BiQuad_32sc(int numBq, int*
   pBufferSize);


IppStatus ippsIIRGetStateSize_BiQuad_32f(int numBq, int* pBufferSize);
IppStatus ippsIIRGetStateSize64f_BiQuad_32f(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize_BiQuad_64f(int numBq, int* pBufferSize);


IppStatus ippsIIRGetStateSize_BiQuad_32fc(int numBq, int* pBufferSize);
IppStatus ippsIIRGetStateSize64fc_BiQuad_32fc(int numBq, int*
   pBufferSize);
IppStatus ippsIIRGetStateSize_BiQuad_64fc(int order, int* pBufferSize);
```

IIRSetTaps

Sets the taps in an IIR filter state.

```
IppStatus ippsIIRSetTaps_32f(const Ipp32f* pTaps, IppsIIRState_32f*
   pState);
IppStatus ippsIIRSetTaps_32fc(const Ipp32fc* pTaps, IppsIIRState_32fc*
   pState);
IppStatus ippsIIRSetTaps_64f(const Ipp64f* pTaps, IppsIIRState_64f*
   pState);
IppStatus ippsIIRSetTaps_64fc(const Ipp64fc* pTaps, IppsIIRState_64fc*
   pState);
IppStatus ippsIIRSetTaps32s_16s(const Ipp32s* pTaps,
   IppsIIRState32s_16s* pState, int tapsFactor);
IppStatus ippsIIRSetTaps32s_16s32f(const Ipp32f* pTaps,
   IppsIIRState32s_16s* pState);
IppStatus ippsIIRSetTaps32sc_16sc(const Ipp32sc* pTaps,
   IppsIIRState32sc_16sc* pState, int tapsFactor);
IppStatus ippsIIRSetTaps32sc_16sc32fc(const Ipp32fc* pTaps,
   IppsIIRState32sc_16sc* pState);
IppStatus ippsIIRSetTaps32f_16s(const Ipp32f* pTaps,
   IppsIIRState32f_16s* pState);
IppStatus ippsIIRSetTaps32fc_16sc(const Ipp32fc* pTaps,
   IppsIIRState32fc_16sc* pState);
```

```
IppStatus ippsIIRSetTaps64f_16s(const Ipp64f* pTaps,
   IppsIIRState64f_16s* pState);
IppStatus ippsIIRSetTaps64f_32s(const Ipp64f* pTaps,
   IppsIIRState64f_32s* pState);
IppStatus ippsIIRSetTaps64f_32f(const Ipp64f* pTaps,
   IppsIIRState64f_32f* pState);
IppStatus ippsIIRSetTaps64fc_16sc(const Ipp64fc* pTaps,
   IppsIIRState64fc_16sc* pState);
IppStatus ippsIIRSetTaps64fc_32sc(const Ipp64fc* pTaps,
   IppsIIRState64fc_32sc* pState);
IppStatus ippsIIRSetTaps64fc_32fc(const Ipp64fc* pTaps,
   IppsIIRState64fc_32fc* pState);
```

IIRGetDlyLine

Retrieves the delay line contents from the IIR filter state.

```
IppStatus ippsIIRGetDlyLine32s_16s(IppsIIRState32s_16s* pState, const
   Ipp32s* pDlyLine);
IppStatus ippsIIRGetDlyLine32f_16s(IppsIIRState32f_16s* pState, const
   Ipp32f* pDlyLine);
IppStatus ippsIIRGetDlyLine64f_16s(IppsIIRState64f_16s* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRGetDlyLine64f_32s(IppsIIRState64f_32s* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRGetDlyLine32sc_16sc(IppsIIRState32sc_16sc* pState,
   const Ipp32sc* pDlyLine);
IppStatus ippsIIRGetDlyLine32fc_16sc(IppsIIRState32fc_16sc* pState,
   const Ipp32fc* pDlyLine);
IppStatus ippsIIRGetDlyLine64fc_16sc(IppsIIRState64fc_16sc* pState,
   const Ipp64fc* pDlyLine);
IppStatus ippsIIRGetDlyLine64fc_32sc(IppsIIRState64fc_32sc* pState,
   const Ipp64fc* pDlyLine);


IppStatus ippsIIRGetDlyLine_32f(IppsIIRState_32f* pState, const Ipp32f*
   pDlyLine);
IppStatus ippsIIRGetDlyLine64f_32f(IppsIIRState64f_32f* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRGetDlyLine_64f(IppsIIRState_64f* pState, const Ipp64f*
   pDlyLine);
IppStatus ippsIIRGetDlyLine_32fc(IppsIIRState_32fc* pState, const
   Ipp32fc* pDlyLine);
IppStatus ippsIIRGetDlyLine64fc_32fc(IppsIIRState64fc_32fc* pState,
   const Ipp64fc* pDlyLine);
```

```
IppStatus ippsIIRGetDlyLine_64fc(IppsIIRState_64fc* pState, const
   Ipp64fc* pDlyLine);
```

IIRSetDlyLine

Sets the delay line contents in an IIR filter state.

```
IppStatus ippsIIRSetDlyLine32s_16s(IppsIIRState32s_16s* pState, const
   Ipp32s* pDlyLine);
IppStatus ippsIIRSetDlyLine32f_16s(IppsIIRState32f_16s* pState, const
   Ipp32f* pDlyLine);
IppStatus ippsIIRSetDlyLine64f_16s(IppsIIRState64f_16s* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRSetDlyLine64f_32s(IppsIIRState64f_32s* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRSetDlyLine32sc_16sc(IppsIIRState32sc_16sc* pState,
   const Ipp32sc* pDlyLine);
IppStatus ippsIIRSetDlyLine32fc_16sc(IppsIIRState32fc_16sc* pState,
   const Ipp32fc* pDlyLine);
IppStatus ippsIIRSetDlyLine64fc_16sc(IppsIIRState64fc_16sc* pState,
   const Ipp64fc* pDlyLine);
IppStatus ippsIIRSetDlyLine64fc_32sc(IppsIIRState64fc_32sc* pState,
   const Ipp64fc* pDlyLine);


IppStatus ippsIIRSetDlyLine_32f(IppsIIRState_32f* pState, const Ipp32f*
   pDlyLine);
IppStatus ippsIIRSetDlyLine64f_32f(IppsIIRState64f_32f* pState, const
   Ipp64f* pDlyLine);
IppStatus ippsIIRSetDlyLine_64f(IppsIIRState_64f* pState, const Ipp64f*
   pDlyLine);
IppStatus ippsIIRSetDlyLine_32fc(IppsIIRState_32fc* pState, const
   Ipp32fc* pDlyLine);
IppStatus ippsIIRSetDlyLine64fc_32fc(IppsIIRState64fc_32fc* pState,
   const Ipp64fc* pDlyLine);
IppStatus ippsIIRSetDlyLine_64fc(IppsIIRState_64fc* pState, const
   Ipp64fc* pDlyLine);
```

IIROne

Filters a single sample through an IIR filter.

```
IppStatus ippsIIROne_32f(Ipp32f src, Ipp32f* pDstVal, IppsIIRState_32f*
   pState);
IppStatus ippsIIROne_64f(Ipp64f src, Ipp64f* pDstVal, IppsIIRState_64f*
   pState);
IppStatus ippsIIROne64f_32f(Ipp32f src, Ipp32f* pDstVal,
   IppsIIRState64f_32f* pState);
```

```
IppStatus ippsIIROne_32fc(Ipp32fc src, Ipp32fc* pDstVal,
   IppsIIRState_32fc* pState);
IppStatus ippsIIROne_64fc(Ipp64fc src, Ipp64fc* pDstVal,
   IppsIIRState_64fc* pState);
IppStatus ippsIIROne64fc_32fc(Ipp32fc src, Ipp32fc* pDstVal,
   IppsIIRState64fc_32fc* pState);
IppStatus ippsIIROne32s_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsIIRState32s_16s* pState, int scaleFactor);
IppStatus ippsIIROne32f_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsIIRState32f_16s* pState, int scaleFactor);
IppStatus ippsIIROne64f_16s_Sfs(Ipp16s src, Ipp16s* pDstVal,
   IppsIIRState64f_16s* pState, int scaleFactor);
IppStatus ippsIIROne64f_32s_Sfs(Ipp32s src, Ipp32s* pDstVal,
   IppsIIRState64f_32s* pState, int scaleFactor);

IppStatus ippsIIROne32sc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsIIRState32sc_16sc* pState, int scaleFactor);
IppStatus ippsIIROne32fc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsIIRState32fc_16sc* pState, int scaleFactor);
IppStatus ippsIIROne64fc_16sc_Sfs(Ipp16sc src, Ipp16sc* pDstVal,
   IppsIIRState64fc_16sc* pState, int scaleFactor);
IppStatus ippsIIROne64fc_32sc_Sfs(Ipp32sc src, Ipp32sc* pDstVal,
   IppsIIRState64fc_32sc* pState, int scaleFactor);
```

IIR

Filters a block of samples through an IIR filter.

```
IppStatus ippsIIR32s_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   IppsIIRState32s_16s* pState, int scaleFactor);
IppStatus ippsIIR32f_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   IppsIIRState32f_16s* pState, int scaleFactor);
IppStatus ippsIIR64f_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   IppsIIRState64f_16s* pState, int scaleFactor);
IppStatus ippsIIR64f_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, int len,
   IppsIIRState64f_32s* pState, int scaleFactor);

IppStatus ippsIIR32sc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, IppsIIRState32sc_16sc* pState, int scaleFactor);
IppStatus ippsIIR32fc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, IppsIIRState32fc_16sc* pState, int scaleFactor);
IppStatus ippsIIR64fc_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst, int
   len, IppsIIRState64fc_16sc* pState, int scaleFactor);
IppStatus ippsIIR64fc_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc* pDst, int
   len, IppsIIRState64fc_32sc* pState, int scaleFactor);
```

```
IppStatus ippsIIR_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   IppsIIRState_32f* pState);
IppStatus ippsIIR_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   IppsIIRState_64f* pState);
IppStatus ippsIIR64f_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   IppsIIRState64f_32f* pState);
IppStatus ippsIIR_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   IppsIIRState_32fc* pState);
IppStatus ippsIIR_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, int len,
   IppsIIRState_64fc* pState);
IppStatus ippsIIR64fc_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, int len,
   IppsIIRState64fc_32fc* pState);
IppStatus ippsIIR32s_16s_ISfs(Ipp16s* pSrcDst, int len,
   IppsIIRState32s_16s* pState, int scaleFactor);
IppStatus ippsIIR32f_16s_ISfs(Ipp16s* pSrcDst, int len,
   IppsIIRState32f_16s* pState, int scaleFactor);
IppStatus ippsIIR64f_16s_ISfs(Ipp16s* pSrcDst, int len,
   IppsIIRState64f_16s* pState, int scaleFactor);
IppStatus ippsIIR64f_32s_ISfs(Ipp32s* pSrcDst, int len,
   IppsIIRState64f_32s* pState, int scaleFactor);


IppStatus ippsIIR32fc_16sc_ISfs(Ipp16sc* pSrcDst, int len,
   IppsIIRState32fc_16sc* pState, int scaleFactor);
IppStatus ippsIIR32sc_16sc_ISfs(Ipp16sc* pSrcDst, int len,
   IppsIIRState32sc_16sc* pState, int scaleFactor);
IppStatus ippsIIR64fc_16sc_ISfs(Ipp16sc* pSrcDst, int len,
   IppsIIRState64fc_16sc* pState, int scaleFactor);
IppStatus ippsIIR64fc_32sc_ISfs(Ipp32sc* pSrcDst, int len,
   IppsIIRState64fc_32sc* pState, int scaleFactor);
IppStatus ippsIIR_32f_I(Ipp32f* pSrcDst, int len, IppsIIRState_32f*
   pState);
IppStatus ippsIIR_64f_I(Ipp64f* pSrcDst, int len, IppsIIRState_64f*
   pState);
IppStatus ippsIIR64f_32f_I(Ipp32f* pSrcDst, int len,
   IppsIIRState64f_32f* pState);
IppStatus ippsIIR_32fc_I(Ipp32fc* pSrcDst, int len, IppsIIRState_32fc*
   pState);
IppStatus ippsIIR_64fc_I(Ipp64fc* pSrcDst, int len, IppsIIRState_64fc*
   pState);
IppStatus ippsIIR64fc_32fc_I(Ipp32fc* pSrcDst, int len,
   IppsIIRState64fc_32fc* pState);
```

IIROne_Direct

Directly filters a single sample through the arbitrary IIR filter.

```
IppStatus ippsIIROne_Direct_16s(Ipp16s src, Ipp16s* pDtsVal, const
   Ipp16s* pTaps, int order, Ipp32s* pDlyLine);
IppStatus ippsIIROne_Direct_16s_I(Ipp16s* pSrcDtsVal, const Ipp16s*
   pTaps, int order, Ipp32s* pDlyLine);
```

IIROne_BiQuadDirect

Directly filters a single sample through the biquad IIR filter.

```
IppStatus ippsIIROne_BiQuadDirect_16s(Ipp16s src, Ipp16s* pDtsVal, const
   Ipp16s* pTaps, int numBq, Ipp32s* pDlyLine);
IppStatus ippsIIROne_BiQuadDirect_16s_I(Ipp16s* pSrcDtsVal, const
   Ipp16s* pTaps, int numBq, Ipp32s* pDlyLine);
```

IIR_Direct

Directly filters a block of samples through the arbitrary IIR filter.

```
IppStatus ippsIIR_Direct_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   const Ipp16s* pTaps, int order, Ipp32s* pDlyLine);
IppStatus ippsIIR_Direct_16s_I(Ipp16s* pSrcDst, int len, const Ipp16s*
   pTaps, int order, Ipp32s* pDlyLine);
```

IIR_BiQuadDirect

Directly filters a block of samples through the biquad IIR filter.

```
IppStatus ippsIIR_BiQuadDirect_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, const Ipp16s* pTaps, int numBq, Ipp32s* pDlyLine);
IppStatus ippsIIR_BiQuadDirect_16s_I(Ipp16s* pSrcDst, int len, const
   Ipp16s* pTaps, int numBq, Ipp32s* pDlyLine);
```

IIRSparseInit

Initializes a sparse IIR filter structure.

```
IppStatus ippsIIRSparseInit_32f(IppsIIRSparseState_32f** ppState, const
   Ipp32f* pNZTaps, const Ipp32s* pNZTapPos, int nzTapsLen1, int
   nzTapsLen2, const Ipp32f* pDlyLine, Ipp8u* pBuffer);
```

IIRSparseGetStateSize

Computes the size of the external buffer for the sparse IIR filter structure.

```
IppStatus ippsIIRSparseGetStateSize_32f(int nzTapsLen1, int nzTapsLen2,
   int order1, int order2, int* pStateSize);
```

IIRSparse

Filters a block of samples through a sparse IIR filter.

```
IppStatus ippsIIRSparse_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   IppsIIRSparseState_32f* pState);
```

FilterMedian

Computes median values for each input array element.

```
IppStatus ippsFilterMedian_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len,
   int maskSize);
IppStatus ippsFilterMedian_16s(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   int maskSize);
IppStatus ippsFilterMedian_32s(const Ipp32s* pSrc, Ipp32s* pDst, int len,
   int maskSize);
IppStatus ippsFilterMedian_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   int maskSize);
IppStatus ippsFilterMedian_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len,
   int maskSize);

IppStatus ippsFilterMedian_8u_I(Ipp8u* pSrcDst, int len, int maskSize);
IppStatus ippsFilterMedian_16s_I(Ipp16s* pSrcDst, int len, int
   maskSize);
IppStatus ippsFilterMedian_32s_I(Ipp32s* pSrcDst, int len, int
   maskSize);
IppStatus ippsFilterMedian_32f_I(Ipp32f* pSrcDst, int len, int
   maskSize);
IppStatus ippsFilterMedian_64f_I(Ipp64f* pSrcDst, int len, int
   maskSize);
```

Transform Functions

Fourier Transform Functions

ConjPack

Converts the data in `Pack` format to complex data format.

```
IppStatus ippsConjPack_16sc(const Ipp16s* pSrc, Ipp16sc* pDst, int
   lenDst);
IppStatus ippsConjPack_32fc(const Ipp32f* pSrc, Ipp32fc* pDst, int
   lenDst);
IppStatus ippsConjPack_64fc(const Ipp64f* pSrc, Ipp64fc* pDst, int
   lenDst);
IppStatus ippsConjPack_16sc_I(Ipp16sc* pSrcDst, int lenDst);
IppStatus ippsConjPack_32fc_I(Ipp32fc* pSrcDst, int lenDst);
IppStatus ippsConjPack_64fc_I(Ipp64fc* pSrcDst, int lenDst);
```

ConjPerm

Converts the data in Perm format to complex data format.

```
IppStatus ippsConjPerm_16sc(const Ipp16s* pSrc, Ipp16sc* pDst, int
   lenDst);
IppStatus ippsConjPerm_32fc(const Ipp32f* pSrc, Ipp32fc* pDst, int
   lenDst);
IppStatus ippsConjPerm_64fc(const Ipp64f* pSrc, Ipp64fc* pDst, int
   lenDst);
IppStatus ippsConjPerm_16sc_I(Ipp16sc* pSrcDst, int lenDst);
IppStatus ippsConjPerm_32fc_I(Ipp32fc* pSrcDst, int lenDst);
IppStatus ippsConjPerm_64fc_I(Ipp64fc* pSrcDst, int lenDst);
```

ConjCcs

Converts the data in CCS format to complex data format.

```
IppStatus ippsConjCcs_16sc(const Ipp16s* pSrc, Ipp16sc* pDst, int
   lenDst);
IppStatus ippsConjCcs_32fc(const Ipp32f* pSrc, Ipp32fc* pDst, int
   lenDst);
IppStatus ippsConjCcs_64fc(const Ipp64f* pSrc, Ipp64fc* pDst, int
   lenDst);
IppStatus ippsConjCcs_16sc_I(Ipp16sc* pSrcDst, int lenDst);
IppStatus ippsConjCcs_32fc_I(Ipp32fc* pSrcDst, int lenDst);
IppStatus ippsConjCcs_64fc_I(Ipp64fc* pSrcDst, int lenDst);
```

MulPack

Multiply the elements of two vectors stored in Pack format.

```
IppStatus ippsMulPack_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int length);
IppStatus ippsMulPack_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int length);
IppStatus ippsMulPack_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int length, int scaleFactor);
IppStatus ippsMulPack_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int
   length);
IppStatus ippsMulPack_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int
   length);
IppStatus ippsMulPack_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   length, int scaleFactor);
```

MulPerm

Multiply the elements of two vectors stored in Perm format.

```
IppStatus ippsMulPerm_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int length);
IppStatus ippsMulPerm_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int length);
IppStatus ippsMulPerm_16s_Sfs(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int length, int scaleFactor);
IppStatus ippsMulPerm_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int
   length);
IppStatus ippsMulPerm_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int
   length);
IppStatus ippsMulPerm_16s_ISfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   length, int scaleFactor);
```

MulPackConj

Multiplies elements of a vector by the elements of a complex conjugate vector stored in Pack format.

```
IppStatus ippsMulPackConj_32f_I(const Ipp32f* pSrc, Ipp32f* pSrcDst, int
   length);
IppStatus ippsMulPackConj_64f_I(const Ipp64f* pSrc, Ipp64f* pSrcDst, int
   length);
```

FFTInitAlloc_R, FFTInitAlloc_C

Allocates memory and initializes an FFT specification structure for real and complex signals.

```
IppStatus ippsFFTInitAlloc_R_16s(IppsFFTSpec_R_16s** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_R_32s(IppsFFTSpec_R_32s** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_R_32f(IppsFFTSpec_R_32f** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_R_64f(IppsFFTSpec_R_64f** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_R_16s32s(IppsFFTSpec_R_16s32s** ppFFTSpec,
   int order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_16s(IppsFFTSpec_C_16s** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_32s(IppsFFTSpec_C_32s** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_32f(IppsFFTSpec_C_32f** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_64f(IppsFFTSpec_C_64f** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
```

```
IppStatus ippsFFTInitAlloc_C_16sc(IppsFFTSpec_C_16sc** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_32sc(IppsFFTSpec_C_32sc** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_32fc(IppsFFTSpec_C_32fc** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
IppStatus ippsFFTInitAlloc_C_64fc(IppsFFTSpec_C_64fc** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint);
```

FFTFree_R, FFTFree_C

Closes an FFT specification structure for real and complex signals.

```
IppStatus ippsFFTFree_R_16s(IppsFFTSpec_R_16s* pFFTSpec);
IppStatus ippsFFTFree_R_32s(IppsFFTSpec_R_32s* pFFTSpec);
IppStatus ippsFFTFree_R_32f(IppsFFTSpec_R_32f* pFFTSpec);
IppStatus ippsFFTFree_R_64f(IppsFFTSpec_R_64f* pFFTSpec);
IppStatus ippsFFTFree_R_16s32s(IppsFFTSpec_R_16s32s* pFFTSpec);
IppStatus ippsFFTFree_C_16s(IppsFFTSpec_C_16s* pFFTSpec);
IppStatus ippsFFTFree_C_32s(IppsFFTSpec_C_32s* pFFTSpec);
IppStatus ippsFFTFree_C_32f(IppsFFTSpec_C_32f* pFFTSpec);
IppStatus ippsFFTFree_C_64f(IppsFFTSpec_C_64f* pFFTSpec);

IppStatus ippsFFTFree_C_16sc(IppsFFTSpec_C_16sc* pFFTSpec);
IppStatus ippsFFTFree_C_32sc(IppsFFTSpec_C_32sc* pFFTSpec);
IppStatus ippsFFTFree_C_32fc(IppsFFTSpec_C_32fc* pFFTSpec);
IppStatus ippsFFTFree_C_64fc(IppsFFTSpec_C_64fc* pFFTSpec);
```

FFTInit_R, FFTInit_C

Initializes an FFT specification structure for real and complex signals.

```
IppStatus ippsFFTInit_R_16s(IppsFFTSpec_R_16s** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_R_32s(IppsFFTSpec_R_32s** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_R_32f(IppsFFTSpec_R_32f** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_R_64f(IppsFFTSpec_R_64f** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
```

```
IppStatus ippsFFTInit_R_16s32s(IppsFFTSpec_R_16s32s** ppFFTSpec, int
   order, int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u*
   pBufInit);
IppStatus ippsFFTInit_C_16s(IppsFFTSpec_C_16s** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_32s(IppsFFTSpec_C_32s** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_32f(IppsFFTSpec_C_32f** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_64f(IppsFFTSpec_C_64f** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);


IppStatus ippsFFTInit_C_16sc(IppsFFTSpec_C_16sc** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_32sc(IppsFFTSpec_C_32sc** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_32fc(IppsFFTSpec_C_32fc** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
IppStatus ippsFFTInit_C_64fc(IppsFFTSpec_C_64fc** ppFFTSpec, int order,
   int flag, IppHintAlgorithm hint, Ipp8u* pMemSpec, Ipp8u* pBufInit);
```

FFTGetSize_R, FFTGetSize_C

Computes sizes of the FFT Specification structure and required working buffers.

```
IppStatus, ippsFFTGetSize_R_16s(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_R_32s(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_R_32f(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_R_64f(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_R_16s32s(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus, ippsFFTGetSize_C_16s(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_32s(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_32f(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_64f(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
```

```
IppStatus ippsFFTGetSize_C_16sc(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_32sc(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_32fc(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
IppStatus ippsFFTGetSize_C_64fc(int order, int flag, IppHintAlgorithm
   hint, int* pSizeSpec, int* pSizeInit, int* pSizeBuf);
```

FFTGetBufSize_R, FFTGetBufSize_C

Computes the size of the FFT work buffer.

```
IppStatus ippsFFTGetBufSize_R_16s(const IppsFFTSpec_R_16s* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_R_32s(const IppsFFTSpec_R_32s* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_R_32f(const IppsFFTSpec_R_32f* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_R_64f(const IppsFFTSpec_R_64f* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_R_16s32s(const IppsFFTSpec_R_16s32s*
   pFFTSpec, int* pSize);


IppStatus ippsFFTGetBufSize_C_16s(const IppsFFTSpec_C_16s* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_32s(const IppsFFTSpec_C_32s* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_32f(const IppsFFTSpec_C_32f* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_64f(const IppsFFTSpec_C_64f* pFFTSpec,
   int* pSize);


IppStatus ippsFFTGetBufSize_C_16sc(const IppsFFTSpec_C_16sc* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_32sc(const IppsFFTSpec_C_32sc* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_32fc(const IppsFFTSpec_C_32fc* pFFTSpec,
   int* pSize);
IppStatus ippsFFTGetBufSize_C_64fc(const IppsFFTSpec_C_64fc* pFFTSpec,
   int* pSize);
```

FFTFwd_CToC

Computes the forward fast Fourier transform (FFT) of a complex signal.

```
IppStatus ippsFFTFwd_CToC_32f(const Ipp32f* pSrcRe, const Ipp32f*
   pSrcIm, Ipp32f* pDstRe, Ipp32f* pDstIm, const IppsFFTSpec_C_32f*
   pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_64f(const Ipp64f* pSrcRe, const Ipp64f*
   pSrcIm, Ipp64f* pDstRe, Ipp64f* pDstIm, const IppsFFTSpec_C_64f*
   pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp16s* pDstRe, Ipp16s* pDstIm, const IppsFFTSpec_C_16s*
   pFFTSpecx, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32s_Sfs(const Ipp32s* pSrcRe, const Ipp32s*
   pSrcIm, Ipp32s* pDstRe, Ipp32s* pDstIm, const IppsFFTSpec_C_32s*
   pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, const
   IppsFFTSpec_C_32fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, const
   IppsFFTSpec_C_64fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst,
   const IppsFFTSpec_C_16sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc* pDst,
   const IppsFFTSpec_C_32sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32f_I(Ipp32f* pSrcDstRe, Ipp32f* pSrcDstIm,
   const IppsFFTSpec_C_32f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_64f_I(Ipp64f* pSrcDstRe, Ipp64f* pSrcDstIm,
   const IppsFFTSpec_C_64f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_16s_ISfs(Ipp16s* pSrcDstRe, Ipp16s* pSrcDstIm,
   const IppsFFTSpec_C_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32s_ISfs(Ipp32s* pSrcDstRe, Ipp32s* pSrcDstIm,
   const IppsFFTSpec_C_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32fc_I(Ipp32fc* pSrcDst, const
   IppsFFTSpec_C_32fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_64fc_I(Ipp64fc* pSrcDst, const
   IppsFFTSpec_C_64fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_16sc_ISfs(Ipp16sc* pSrcDst, const
   IppsFFTSpec_C_16sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_CToC_32sc_ISfs(Ipp32sc* pSrcDst, const
   IppsFFTSpec_C_32sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

FFTInv_CToC

Computes the inverse fast Fourier transform (FFT) of a complex signal.

```
IppStatus ippsFFTInv_CToC_32f(const Ipp32f* pSrcRe, const Ipp32f*
   pSrcIm, Ipp32f* pDstRe, Ipp32f* pDstIm, const IppsFFTSpec_C_32f*
   pFFTSpec, Ipp8u* pBuffer);
```

```
IppStatus ippsFFTInv_CToC_64f(const Ipp64f* pSrcRe, const Ipp64f*
   pSrcIm, Ipp64f* pDstRe, Ipp64f* pDstIm, const IppsFFTSpec_C_64f*
   pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_16s_Sfs(const Ipp16s* pSrcRe,
   const Ipp16s* pSrcIm, Ipp16s* pDstRe, Ipp16s* pDstIm,
   const IppsFFTSpec_C_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32s_Sfs(const Ipp32s* pSrcRe, const Ipp32s*
   pSrcIm, Ipp32s* pDstRe, Ipp32s* pDstIm, const IppsFFTSpec_C_32s*
   pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, const
   IppsFFTSpec_C_32fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, const
   IppsFFTSpec_C_64fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst,
   const IppsFFTSpec_C_16sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32sc_Sfs(const Ipp32sc* pSrc, Ipp32sc* pDst,
   const IppsFFTSpec_C_32sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32f_I(Ipp32f* pSrcDstRe, Ipp32f* pSrcDstIm,
   const IppsFFTSpec_C_32f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_64f_I(Ipp64f* pSrcDstRe, Ipp64f* pSrcDstIm,
   const IppsFFTSpec_C_64f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_16s_ISfs(Ipp16s* pSrcDstRe, Ipp16s* pSrcDstIm,
   const IppsFFTSpec_C_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32s_ISfs(Ipp32s* pSrcDstRe, Ipp32s* pSrcDstIm,
   const IppsFFTSpec_C_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32fc_I(Ipp32fc* pSrcDst, const
   IppsFFTSpec_C_32fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_64fc_I(Ipp64fc* pSrcDst, const
   IppsFFTSpec_C_64fc* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_16sc_ISfs(Ipp16sc* pSrcDst, const
   IppsFFTSpec_C_16sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);

IppStatus ippsFFTInv_CToC_32sc_ISfs(Ipp32sc* pSrcDst, const
   IppsFFTSpec_C_32sc* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

FFTFwd_RToPack, FFTFwd_RToPerm, FFTFwd_RToCCS

Computes the forward or inverse fast Fourier transform (FFT) of a real signal.

```
IppStatus ippsFFTFwd_RToPack_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_RToPack_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);

IppStatus ippsFFTFwd_RToPack_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

```
IppStatus ippsFFTFwd_RToPack_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPack_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPack_64f_I(Ipp64f* pSrcDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPack_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPack_32s_ISfs(Ipp32s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_64f_I(Ipp64f* pSrcDst const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToPerm_32s_ISfs(Ipp32s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_16s32s_Sfs(const Ipp16s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_16s32s* pFFTSpec, int scaleFactor, Ipp8u*
   pBuffer);
IppStatus ippsFFTFwd_RToCCS_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_64f_I(Ipp64f* pSrcDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTFwd_RToCCS_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

```
IppStatus ippsFFTFwd_RToCCS_32s_ISfs(Ipp32s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

FFTInv_PackToR, FFTInv_PermToR, FFTInv_CCSToR

Computes the inverse fast Fourier transform (FFT) of a real signal.

```
IppStatus ippsFFTInv_PackToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_64f_I(Ipp64f* pSrcDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PackToR_32s_ISfs(Ipp32s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_64f_I(Ipp64f* pSrcDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_PermToR_32s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
```

```
IppStatus ippsFFTInv_CCSToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst,
   const IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst,
   const IppsFFTSpec_R_16s32s* pFFTSpec, int scaleFactor, Ipp8u*
   pBuffer);
IppStatus ippsFFTInv_CCSToR_32f_I(Ipp32f* pSrcDst, const
   IppsFFTSpec_R_32f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_64f_I(Ipp64f* pSrcDst, const
   IppsFFTSpec_R_64f* pFFTSpec, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_16s_ISfs(Ipp16s* pSrcDst, const
   IppsFFTSpec_R_16s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsFFTInv_CCSToR_32s_ISfs(Ipp32s* pSrcDst, const
   IppsFFTSpec_R_32s* pFFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

DFTInitAlloc_R, DFTInitAlloc_C

Initializes the DFT specification structure for real and complex signals.

```
IppStatus ippsDFTInitAlloc_R_16s(IppsDFTSpec_R_16s** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_R_32f(IppsDFTSpec_R_32f** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_R_64f(IppsDFTSpec_R_64f** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_C_16s(IppsDFTSpec_C_16s** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_C_32f(IppsDFTSpec_C_32f** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_C_64f(IppsDFTSpec_C_64f** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);


IppStatus ippsDFTInitAlloc_C_16sc(IppsDFTSpec_C_16sc** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_C_32fc(IppsDFTSpec_C_32fc** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTInitAlloc_C_64fc(IppsDFTSpec_C_64fc** pDFTSpec, int
   length, int flag, IppHintAlgorithm hint);
```

DFTFree_R, DFTFree_C

Closes the DFT specification structure for real and complex signals.

```
IppStatus ippsDFTFree_R_16s(IppsDFTSpec_R_16s* pDFTSpec);
IppStatus ippsDFTFree_R_32f(IppsDFTSpec_R_32f* pDFTSpec);
```

```
IppStatus ippsDFTFree_R_64f(IppsDFTSpec_R_64f* pDFTSpec);
IppStatus ippsDFTFree_C_16s(IppsDFTSpec_C_16s* pDFTSpec);
IppStatus ippsDFTFree_C_32f(IppsDFTSpec_C_32f* pDFTSpec);
IppStatus ippsDFTFree_C_64f(IppsDFTSpec_C_64f* pDFTSpec);

IppStatus ippsDFTFree_C_16sc(IppsDFTSpec_C_16sc* pDFTSpec);
IppStatus ippsDFTFree_C_32fc(IppsDFTSpec_C_32fc* pDFTSpec);
IppStatus ippsDFTFree_C_64fc(IppsDFTSpec_C_64fc* pDFTSpec);
```

DFTGetBufSize_R, DFTGetBufSize_C

Computes the size of the DFT work buffer.

```
IppStatus ippsDFTGetBufSize_R_16s(const IppsDFTSpec_R_16s*  pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_R_32f(const IppsDFTSpec_R_32f*  pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_R_64f(const IppsDFTSpec_R_64f*  pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_C_16s(const IppsDFTSpec_C_16s*  pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_C_32f(const IppsDFTSpec_C_32f*  pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_C_64f(const IppsDFTSpec_C_64f*  pDFTSpec,
   int* pSize);

IppStatus ippsDFTGetBufSize_C_16sc(const IppsDFTSpec_C_16sc* pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_C_32fc(const IppsDFTSpec_C_32fc* pDFTSpec,
   int* pSize);
IppStatus ippsDFTGetBufSize_C_64fc(const IppsDFTSpec_C_64fc* pDFTSpec,
   int* pSize);
```

DFTFwd_CToC

Computes the forward discrete Fourier transform of a complex signal.

```
IppStatus ippsDFTFwd_CToC_32f(const Ipp32f* pSrcRe, const Ipp32f*
   pSrcIm, Ipp32f* pDstRe, Ipp32f* pDstIm, const IppsDFTSpec_C_32f*
   pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_CToC_64f(const Ipp64f* pSrcRe, const Ipp64f*
   pSrcIm, Ipp64f* pDstRe, Ipp64f* pDstIm, const IppsDFTSpec_C_64f*
   pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_CToC_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp16s* pDstRe, Ipp16s* pDstIm, const IppsDFTSpec_C_16s*
   pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

```
IppStatus ippsDFTFwd_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, const
   IppsDFTSpec_C_32fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst, const
   IppsDFTSpec_C_64fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_CToC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst,
   const IppsDFTSpec_C_16sc* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

DFTInv_CToC

Computes the inverse discrete Fourier transform of a complex signal.

```
IppStatus ippsDFTInv_CToC_32f(const Ipp32f* pSrcRe, const Ipp32f*
   pSrcIm, Ipp32f* pDstRe, Ipp32f* pDstIm, const IppsDFTSpec_C_32f*
   pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CToC_64f(const Ipp64f* pSrcRe, const Ipp64f*
   pSrcIm, Ipp64f* pDstRe, Ipp64f* pDstIm, const IppsDFTSpec_C_64f*
   pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CToC_16s_Sfs(const Ipp16s* pSrcRe, const Ipp16s*
   pSrcIm, Ipp16s* pDstRe, Ipp16s* pDstIm, const IppsDFTSpec_C_16s*
   pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst, const
   IppsDFTSpec_C_32fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
   const IppsDFTSpec_C_64fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CToC_16sc_Sfs(const Ipp16sc* pSrc, Ipp16sc* pDst,
   const IppsDFTSpec_C_16sc* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

DFTFwd_RToPack, DFTFwd_RToPerm, DFTFwd_RToCCS

Computes the forward discrete Fourier transform of a real signal.

```
IppStatus ippsDFTFwd_RToPack_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToPack_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToPack_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToPerm_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToPerm_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToPerm_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToCCS_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
```

```
IppStatus ippsDFTFwd_RToCCS_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTFwd_RToCCS_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

DFTInv_PackToR, DFTInv_PermToR, DFTInv_CCSToR

Computes the inverse discrete Fourier transform of a real signal.

```
IppStatus ippsDFTInv_PackToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_PackToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_PackToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDFTInv_PermToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_PermToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_PermToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CCSToR_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDFTSpec_R_32f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CCSToR_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDFTSpec_R_64f* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTInv_CCSToR_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   const IppsDFTSpec_R_16s* pDFTSpec, int scaleFactor, Ipp8u* pBuffer);
```

DFTOutOrdInitAlloc_C

Initializes the out-of-order discrete Fourier transform structure.

```
IppStatus ippsDFTOutOrdInitAlloc_C_32fc(IppsDFTOutOrdSpec_C_32fc**
   pDFTSpec, int length, int flag, IppHintAlgorithm hint);
IppStatus ippsDFTOutOrdInitAlloc_C_64fc(IppsDFTOutOrdSpec_C_64fc**
   pDFTSpec, int length, int flag, IppHintAlgorithm hint);
```

DFTOutOrdFree_C

Closes the out-of-order discrete Fourier transform structure.

```
IppStatus ippsDFTOutOrdFree_C_32fc(IppsDFTOutOrdSpec_C_32fc* pDFTSpec);
IppStatus ippsDFTOutOrdFree_C_64fc(IppsDFTOutOrdSpec_C_64fc* pDFTSpec);
```

DFTOutOrdGetBufSize_C

Computes the size of the work buffer for out-of-order discrete Fourier transform.

```
IppStatus ippsDFTOutOrdGetBufSize_C_32fc(const IppsDFTOutOrdSpec_C_32fc*
   pDFTSpec, int* pSize);
```

```
IppStatus ippsDFTOutOrdGetBufSize_C_64fc(const IppsDFTOutOrdSpec_C_64fc*
   pDFTSpec, int* pSize);
```

DFTOutOrdFwd_CToC

Computes the forward out-of-order discrete Fourier transform.

```
IppStatus ippsDFTOutOrdFwd_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
   const IppsDFTOutOrdSpec_C_32fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTOutOrdFwd_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
   const IppsDFTOutOrdSpec_C_64fc* pDFTSpec, Ipp8u* pBuffer);
```

DFTOutOrdInv_CToC

Computes the inverse out-of-order discrete Fourier transform.

```
IppStatus ippsDFTOutOrdInv_CToC_32fc(const Ipp32fc* pSrc, Ipp32fc* pDst,
   const IppsDFTOutOrdSpec_C_32fc* pDFTSpec, Ipp8u* pBuffer);
IppStatus ippsDFTOutOrdInv_CToC_64fc(const Ipp64fc* pSrc, Ipp64fc* pDst,
   const IppsDFTOutOrdSpec_C_64fc* pDFTSpec, Ipp8u* pBuffer);
```

Goertz

Computes the discrete Fourier transform for a given frequency for a single complex signal.

```
IppStatus ippsGoertz_32f(const Ipp32f* pSrc, int len, Ipp32fc* pVal,
   Ipp32f rFreq);
IppStatus ippsGoertz_32fc(const Ipp32fc* pSrc, int len, Ipp32fc* pVal,
   Ipp32f rFreq);
IppStatus ippsGoertz_64fc(const Ipp64fc* pSrc, int len, Ipp64fc* pVal,
   Ipp64f rFreq);
IppStatus ippsGoertz_16s_Sfs(const Ipp16s* pSrc, int len, Ipp16sc* pVal,
   Ipp32f rFreq, int scaleFactor);
IppStatus ippsGoertz_16sc_Sfs(const Ipp16sc* pSrc, int len, Ipp16sc*
   pVal, Ipp32f rFreq, int scaleFactor);
IppStatus ippsGoertzQ15_16sc_Sfs(const Ipp16sc* pSrc, int len, Ipp16sc*
   pVal, Ipp16s rFreqQ15, int scaleFactor);
```

GoertzTwo

Computes two discrete Fourier transforms for a given frequency for a single complex signal.

```
IppStatus ippsGoertzTwo_32fc(const Ipp32fc* pSrc, int len, Ipp32fc
   pVal[2], const Ipp32f rFreq[2]);
IppStatus ippsGoertzTwo_64fc(const Ipp64fc* pSrc, int len, Ipp64fc
   pVal[2], const Ipp64f rFreq[2]);
IppStatus ippsGoertzTwo_16sc_Sfs(const Ipp16sc* pSrc, int len, Ipp16sc
   pVal[2], const Ipp32f rFreq[2], int scaleFactor);
IppStatus ippsGoertzTwoQ15_16sc_Sfs(const Ipp16sc* pSrc, int len,
   Ipp16sc pVal[2], const Ipp16s rFreqQ15[2], int scaleFactor);
```

Discrete Cosine Transform Functions

DCTFwdInitAlloc, DCTInvInitAlloc

Initializes the discrete cosine transform structure.

```
IppStatus ippsDCTFwdInitAlloc_16s(IppsDCTFwdSpec_16s** pDCTSpec, int
   length, IppHintAlgorithm hint);
IppStatus ippsDCTFwdInitAlloc_32f(IppsDCTFwdSpec_32f** pDCTSpec, int
   length, IppHintAlgorithm hint);
IppStatus ippsDCTFwdInitAlloc_64f(IppsDCTFwdSpec_64f** pDCTSpec, int
   length, IppHintAlgorithm hint);
IppStatus ippsDCTInvInitAlloc_16s(IppsDCTInvSpec_16s** pDCTSpec, int
   length, IppHintAlgorithm hint);
IppStatus ippsDCTInvInitAlloc_32f(IppsDCTInvSpec_32f** pDCTSpec, int
   length, IppHintAlgorithm hint);
IppStatus ippsDCTInvInitAlloc_64f(IppsDCTInvSpec_64f** pDCTSpec, int
   length, IppHintAlgorithm hint);
```

DCTFwdFree, DCTInvFree

Closes a discrete cosine transform structure.

```
IppStatus ippsDCTFwdFree_16s(IppsDCTFwdSpec_16s* pDCTSpec);
IppStatus ippsDCTFwdFree_32f(IppsDCTFwdSpec_32f* pDCTSpec);
IppStatus ippsDCTFwdFree_64f(IppsDCTFwdSpec_64f* pDCTSpec);
IppStatus ippsDCTInvFree_16s(IppsDCTInvSpec_16s* pDCTSpec);
IppStatus ippsDCTInvFree_32f(IppsDCTInvSpec_32f* pDCTSpec);
IppStatus ippsDCTInvFree_64f(IppsDCTInvSpec_64f* pDCTSpec);
```

DCTFwdGetBufSize, DCTInvGetBufSize

Computes the size of the DCT work buffer.

```
IppStatus ippsDCTFwdGetBufSize_32f(const IppsDCTFwdSpec_32f* pDCTSpec,
   int* pSize);
IppStatus ippsDCTFwdGetBufSize_16s(const IppsDCTFwdSpec_16s* pDCTSpec,
   int* pSize);
IppStatus ippsDCTFwdGetBufSize_64f(const IppsDCTFwdSpec_64f* pDCTSpec,
   int* pSize);
IppStatus ippsDCTInvGetBufSize_16s(const IppsDCTInvSpec_16s* pDCTSpec,
   int* pSize);
IppStatus ippsDCTInvGetBufSize_32f(const IppsDCTInvSpec_32f* pDCTSpec,
   int* pSize);
IppStatus ippsDCTInvGetBufSize_64f(const IppsDCTInvSpec_64f* pDCTSpec,
   int* pSize);
```

DCTFwd, DCTInv

Computes the forward or inverse discrete cosine transform of a signal.

```
IppStatus ippsDCTFwd_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDCTFwdSpec_32f* pDCTSpec, Ipp8u* pBuffer);
IppStatus ippsDCTFwd_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDCTFwdSpec_64f* pDCTSpec, Ipp8u* pBuffer);
IppStatus ippsDCTFwd_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, const
   IppsDCTFwdSpec_16s* pDCTSpec, int scaleFactor, Ipp8u* pBuffer);
IppStatus ippsDCTInv_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDCTInvSpec_32f* pDCTSpec, Ipp8u* pBuffer);
IppStatus ippsDCTInv_64f(const Ipp64f* pSrc, Ipp64f* pDst, const
   IppsDCTInvSpec_64f* pDCTSpec, Ipp8u* pBuffer);
IppStatus ippsDCTInv_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, const
   IppsDCTInvSpec_16s* pDCTSpec, int scaleFactor, Ipp8u* pBuffer);
```

Hilbert Transform Functions

HilbertInitAlloc

Initializes the Hilbert transform structure.

```
IppStatus ippsHilbertInitAlloc_32f32fc(IppsHilbertSpec_32f32fc** pSpec,
   int length, IppHintAlgorithm hint);
IppStatus ippsHilbertInitAlloc_16s32fc(IppsHilbertSpec_16s32fc** pSpec,
   int length, IppHintAlgorithm hint);
IppStatus ippsHilbertInitAlloc_16s16sc(IppsHilbertSpec_16s16sc** pSpec,
   int length, IppHintAlgorithm hint);
```

HilbertFree

Closes a Hilbert transform structure.

```
IppStatus ippsHilbertFree_32f32fc(IppsHilbertSpec_32f32fc* pSpec);
IppStatus ippsHilbertFree_16s32fc(IppsHilbertSpec_16s32fc* pSpec);
IppStatus ippsHilbertFree_16s16sc(IppsHilbertSpec_16s16sc* pSpec);
```

Hilbert

Computes an analytic signal using the Hilbert transform.

```
IppStatus ippsHilbert_32f32fc(const Ipp32f* pSrc, Ipp32fc* pDst,
   IppsHilbertSpec_32f32fc* pSpec);
IppStatus ippsHilbert_16s32fc(const Ipp16s* pSrc, Ipp32fc* pDst,
   IppsHilbertSpec_16s32fc* pSpec);
IppStatus ippsHilbert_16s16sc_Sfs(const Ipp16s* pSrc, Ipp16sc* pDst,
   IppsHilbertSpec_16s16sc* pSpec, int scaleFactor);
```

Wavelet Transform Functions

WTHaarFwd, WTHaarInv

Performs forward or inverse single-level discrete wavelet Haar transforms.

```
IppStatus ippsWTHaarFwd_8s(const Ipp8s* pSrc, int lenSrc, Ipp8s* pDstLow,
   Ipp8s* pDstHigh);
IppStatus ippsWTHaarFwd_16s(const Ipp16s* pSrc, int lenSrc, Ipp16s*
   pDstLow, Ipp16s* pDstHigh);
IppStatus ippsWTHaarFwd_32s(const Ipp32s* pSrc, int lenSrc, Ipp32s*
   pDstLow, Ipp32s* pDstHigh);
IppStatus ippsWTHaarFwd_64s(const Ipp64s* pSrc, int lenSrc, Ipp64s*
   pDstLow, Ipp64s* pDstHigh);
IppStatus ippsWTHaarFwd_32f(const Ipp32f* pSrc, int lenSrc, Ipp32f*
   pDstLow, Ipp32f* pDstHigh);
IppStatus ippsWTHaarFwd_64f(const Ipp64f* pSrc, int lenSrc, Ipp64f*
   pDstLow, Ipp64f* pDstHigh);

IppStatus ippsWTHaarFwd_8s_Sfs(const Ipp8s* pSrc, int lenSrc, Ipp8s*
   pDstLow, Ipp8s* pDstHigh, int scaleFactor);
IppStatus ippsWTHaarFwd_16s_Sfs(const Ipp16s* pSrc, int lenSrc, Ipp16s*
   pDstLow, Ipp16s* pDstHigh, int scaleFactor );
IppStatus ippsWTHaarFwd_32s_Sfs(const Ipp32s* pSrc, int lenSrc, Ipp32s*
   pDstLow, Ipp32s* pDstHigh, int scaleFactor);
IppStatus ippsWTHaarFwd_64s_Sfs(const Ipp64s* pSrc, int lenSrc, Ipp64s*
   pDstLow, Ipp64s* pDstHigh, int scaleFactor);
IppStatus ippsWTHaarInv_8s(const Ipp8s* pSrcLow, const Ipp8s* pSrcHigh,
   Ipp8s* pDst, int lenDst);
IppStatus ippsWTHaarInv_16s(const Ipp16s* pSrcLow, const Ipp16s*
   pSrcHigh, Ipp16s* pDst, int lenDst);
IppStatus ippsWTHaarInv_32s(const Ipp32s* pSrcLow, const Ipp32s*
   pSrcHigh, Ipp32s* pDst, int lenDst);
IppStatus ippsWTHaarInv_64s(const Ipp64s* pSrcLow, const Ipp64s*
   pSrcHigh, Ipp64s* pDst, int lenDst);
IppStatus ippsWTHaarInv_32f(const Ipp32f* pSrcLow, const Ipp32f*
   pSrcHigh, Ipp32f* pDst, int lenDst);
IppStatus ippsWTHaarInv_64f(const Ipp64f* pSrcLow, const Ipp64f*
   pSrcHigh, Ipp64f* pDst, int lenDst);

IppStatus ippsWTHaarInv_8s_Sfs(const Ipp8s* pSrcLow, const Ipp8s*
   pSrcHigh, Ipp8s* pDst, int lenDst, int scaleFactor);
```

```
IppStatus ippsWTHaarInv_16s_Sfs(const Ipp16s* pSrcLow, const Ipp16s*
   pSrcHigh, Ipp16s* pDst, int lenDst, int scaleFactor);
IppStatus ippsWTHaarInv_32s_Sfs(const Ipp32s* pSrcLow, const Ipp32s*
   pSrcHigh, Ipp32s* pDst, int lenDst, int scaleFactor);
IppStatus ippsWTHaarInv_64s_Sfs(const Ipp64s* pSrcLow, const Ipp64s*
   pSrcHigh, Ipp64s* pDst, int lenDst, int scaleFactor);
```

WTFwdInitAlloc, WTInvInitAlloc

Initializes the wavelet transform structure.

```
IppStatus ippsWTFwdInitAlloc_32f(IppsWTFwdState_32f** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTFwdInitAlloc_8s32f(IppsWTFwdState_8s32f** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTFwdInitAlloc_8u32f(IppsWTFwdState_8u32f** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTFwdInitAlloc_16s32f(IppsWTFwdState_16s32f** pState,
   const Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f*
   pTapsHigh, int lenHigh, int offsHigh);
IppStatus ippsWTFwdInitAlloc_16u32f(IppsWTFwdState_16u32f** pState,
   const Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f*
   pTapsHigh, int lenHigh, int offsHigh);
IppStatus ippsWTInvInitAlloc_32f(IppsWTInvState_32f** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTInvInitAlloc_32f8s(IppsWTInvState_32f8s** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTInvInitAlloc_32f8u(IppsWTInvState_32f8u** pState, const
   Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f* pTapsHigh,
   int lenHigh, int offsHigh);
IppStatus ippsWTInvInitAlloc_32f16s(IppsWTInvState_32f16s** pState,
   const Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f*
   pTapsHigh, int lenHigh, int offsHigh);
IppStatus ippsWTInvInitAlloc_32f16u(IppsWTInvState_32f16u** pState,
   const Ipp32f* pTapsLow, int lenLow, int offsLow, const Ipp32f*
   pTapsHigh, int lenHigh, int offsHigh);
```

WTFwdFree, WTInvFree

Closes a wavelet transform structure.

```
IppStatus ippsWTFwdFree_32f(IppsWTFwdState_32f* pState);
IppStatus ippsWTFwdFree_8s32f(IppsWTFwdState_8s32f* pState);
```

```
IppStatus ippsWTFwdFree_8u32f(IppsWTFwdState_8u32f* pState);
IppStatus ippsWTFwdFree_16s32f(IppsWTFwdState_16s32f* pState);
IppStatus ippsWTFwdFree_16u32f(IppsWTFwdState_16u32f* pState);
IppStatus ippsWTInvFree_32f(IppsWTInvState_32f* pState);
IppStatus ippsWTInvFree_32f8s(IppsWTInvState_32f8s* pState);
IppStatus ippsWTInvFree_32f8u(IppsWTInvState_32f8u* pState);
IppStatus ippsWTInvFree_32f16s (IppsWTInvState_32f16s* pState);
IppStatus ippsWTInvFree_32f16u(IppsWTInvState_32f16u* pState);
```

WTFwd

Computes the forward wavelet transform.

```
IppStatus ippsWTFwd_32f(const Ipp32f* pSrc, Ipp32f* pDstLow, Ipp32f*
   pDstHigh, int dstLen, IppsWTFwdState_32f* pState);
IppStatus ippsWTFwd_8s32f(const Ipp8s* pSrc, Ipp32f* pDstLow, Ipp32f*
   pDstHigh, int dstLen, IppsWTFwdState_8s32f* pState);
IppStatus ippsWTFwd_8u32f(const Ipp8u* pSrc, Ipp32f* pDstLow, Ipp32f*
   pDstHigh, int dstLen, IppsWTFwdState_8u32f* pState);
IppStatus ippsWTFwd_16s32f(const Ipp16s* pSrc, Ipp32f* pDstLow, Ipp32f*
   pDstHigh, int dstLen, IppsWTFwdState_16s32f* pState);
IppStatus ippsWTFwd_16u32f(const Ipp16u* pSrc, Ipp32f* pDstLow, Ipp32f*
   pDstHigh, int dstLen, IppsWTFwdState_16u32f* pState);
```

WTFwdSetDlyLine, WTFwdGetDlyLine

Sets and gets the delay lines of the forward wavelet transform.

```
IppStatus ippsWTFwdSetDlyLine_32f(IppsWTFwdState_32f* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTFwdSetDlyLine_8s32f(IppsWTFwdState_8s32f* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTFwdSetDlyLine_8u32f(IppsWTFwdState_8u32f* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTFwdSetDlyLine_16s32f(IppsWTFwdState_16s32f* pState,
   const Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTFwdSetDlyLine_16u32f(IppsWTFwdState_16u32f* pState,
   const Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);


IppStatus ippsWTFwdGetDlyLine_32f(IppsWTFwdState_32f* pState, Ipp32f*
   pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTFwdGetDlyLine_8s32f(IppsWTFwdState_8s32f* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
```

```
IppStatus ippsWTFwdGetDlyLine_8u32f(IppsWTFwdState_8u32f* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTFwdGetDlyLine_16s32f(IppsWTFwdState_16s32f* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTFwdGetDlyLine_16u32f(IppsWTFwdState_16u32f* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
```

WTInv

Computes the inverse wavelet transform.

```
IppStatus ippsWTInv_32f(const Ipp32f* pSrcLow, const Ipp32f* pSrcHigh,
   int srcLen, Ipp32f* pDst, IppsWTInvState_32f* pState);
IppStatus ippsWTInv_32f8s(const Ipp32f* pSrcLow, const Ipp32f* pSrcHigh,
   int srcLen, Ipp8s* pDst, IppsWTInvState_32f8s* pState);
IppStatus ippsWTInv_32f8u(const Ipp32f* pSrcLow, const Ipp32f* pSrcHigh,
   int srcLen, Ipp8u* pDst, IppsWTInvState_32f8u* pState);
IppStatus ippsWTInv_32f16s(const Ipp32f* pSrcLow, const Ipp32f*
   pSrcHigh, int srcLen, Ipp16s* pDst, IppsWTInvState_32f16s* pState);
IppStatus ippsWTInv_32f16u(const Ipp32f* pSrcLow, const Ipp32f*
   pSrcHigh, int srcLen, Ipp16u* pDst, IppsWTInvState_32f16u* pState);
```

WTInvSetDlyLine, WTInvGetDlyLine

Sets and gets the delay lines of the inverse wavelet transform.

```
IppStatus ippsWTInvSetDlyLine_32f(IppsWTInvState_32f* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTInvSetDlyLine_32f8s(IppsWTInvState_32f8s* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTInvSetDlyLine_32f8u(IppsWTInvState_32f8u* pState, const
   Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTInvSetDlyLine_32f16s(IppsWTInvState_32f16s* pState,
   const Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);
IppStatus ippsWTInvSetDlyLine_32f16u(IppsWTInvState_32f16u* pState,
   const Ipp32f* pDlyLow, const Ipp32f* pDlyHigh);


IppStatus ippsWTInvGetDlyLine_32f(IppsWTInvState_32f* pState, Ipp32f*
   pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTInvGetDlyLine_32f8s(IppsWTInvState_32f8s* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTInvGetDlyLine_32f8u(IppsWTInvState_32f8u* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTInvGetDlyLine_32f16s(IppsWTInvState_32f16s* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
IppStatus ippsWTInvGetDlyLine_32f16u(IppsWTInvState_32f16u* pState,
   Ipp32f* pDlyLow, Ipp32f* pDlyHigh);
```

Speech Recognition Functions

Basic Arithmetics

AddAllRowSum

Calculates the sums of column vectors in a matrix and adds the sums to a vector.

```
IppStatus ippsAddAllRowSum_32f_D2(const Ipp32f* pSrc, int step,
   int height, Ipp32f* pSrcDst, int width);
IppStatus ippsAddAllRowSum_32f_D2L(const Ipp32f** mSrc, int height,
   Ipp32f* pSrcDst, int width);
```

SumColumn

Calculates sums of column vectors in a matrix.

```
IppStatus ippsSumColumn_16s32s_D2Sfs(const Ipp16s* pSrc, int step, int
   height, Ipp32s* pDst, int width, int scaleFactor);
IppStatus ippsSumColumn_16s32f_D2(const Ipp16s* pSrc, int step, int
   height, Ipp32f* pDst, int width);
IppStatus ippsSumColumn_32f_D2(const Ipp32f* pSrc, int step, int height,
   Ipp32f* pDst, int width);
IppStatus ippsSumColumn_64f_D2(const Ipp64f* pSrc, int step, int height,
   Ipp64f* pDst, int width);
IppStatus ippsSumColumn_16s32s_D2LSfs(const Ipp16s** mSrc, int height,
   Ipp32s* pDst, int width, int scaleFactor);
IppStatus ippsSumColumn_16s32f_D2L(const Ipp16s** mSrc, int height,
   Ipp32f* pDst, int width);
IppStatus ippsSumColumn_32f_D2L(const Ipp32f** mSrc, int height, Ipp32f*
   pDst, int width);
IppStatus ippsSumColumn_64f_D2L(const Ipp64f** mSrc, int height, Ipp64f*
   pDst, int width);
```

SumRow

Calculates sums of row vectors in a matrix.

```
IppStatus ippsSumRow_16s32s_D2Sfs(const Ipp16s*  pSrc, int width, int
   step, Ipp32s* pDst, int height, int scaleFactor);
IppStatus ippsSumRow_16s32f_D2(const Ipp16s* pSrc, int width, int step,
   Ipp32f* pDst, int height);
IppStatus ippsSumRow_32f_D2(const Ipp32f* pSrc, int width, int step,
   Ipp32f* pDst, int height);
IppStatus ippsSumRow_64f_D2(const Ipp64f* pSrc, int width, int step,
   Ipp64f* pDst, int height);
IppStatus ippsSumRow_16s32s_D2LSfs(const Ipp16s** mSrc, int width,
   Ipp32s* pDst, int height, int scaleFactor);
```

```
IppStatus ippsSumRow_16s32f_D2L(const Ipp16s** mSrc, int width, Ipp32f*
   pDst, int height);
IppStatus ippsSumRow_32f_D2L(const Ipp32f** mSrc, int width, Ipp32f*
   pDst, int height);
IppStatus ippsSumRow_64f_D2L(const Ipp64f** mSrc, int width, Ipp64f*
   pDst, int height);
```

SubRow

Subtracts a vector from all matrix rows.

```
IppStatus ippsSubRow_16s_D2(const Ipp16s* pSrc, int width, Ipp16s*
   pSrcDst, int dstStep, int height);
IppStatus ippsSubRow_32f_D2(const Ipp32f* pSrc, int width, Ipp32f*
   pSrcDst, int dstStep, int height);
IppStatus ippsSubRow_16s_D2L(const Ipp16s* pSrc, Ipp16s** mSrcDst, int
   width, int height);
IppStatus ippsSubRow_32f_D2L(const Ipp32f* pSrc, Ipp32f** mSrcDst, int
   width, int height);
```

CopyColumn_Indirect

Copies the input matrix with columns redirection.

```
IppStatus ippsCopyColumn_Indirect_16s_D2(const Ipp16s* pSrc, int srcLen,
   int srcStep, Ipp16s* pDst, const Ipp32s* pIndx, int dstLen, int
   dstStep, int height);
IppStatus ippsCopyColumn_Indirect_32f_D2(const Ipp32f* pSrc, int srcLen,
   int srcStep, Ipp32f* pDst, const Ipp32s* pIndx, int dstLen, int
   dstStep, int height);
IppStatus ippsCopyColumn_Indirect_64f_D2 (const Ipp64f* pSrc, int
   srcLen, int srcStep, Ipp64f* pDst, const Ipp32s* pIndx, int dstLen,
   int dstStep, int height);
IppStatus ippsCopyColumn_Indirect_16s_D2L(const Ipp16s** mSrc, int
   srcLen, Ipp16s** mDst, const Ipp32s* pIndx, int dstLen, int height);
IppStatus ippsCopyColumn_Indirect_32f_D2L(const Ipp32f** mSrc, int
   srcLen, Ipp32f** mDst, const Ipp32s* pIndx, int dstLen, int height);
IppStatus ippsCopyColumn_Indirect_64f_D2L (const Ipp64f** mSrc, int
   srcLen, Ipp64f** mDst, const Ipp32s* pIndx, int dstLen, int height);
```

BlockDMatrixInitAlloc

Initializes the structure that represents a symmetric block diagonal matrix.

```
IppStatus ippsBlockDMatrixInitAlloc_16s(IppsBlockDMatrix_16s** pMatrix,
   const Ipp16s** mSrc, const int* bSize, int nBlocks);
IppStatus ippsBlockDMatrixInitAlloc_32f(IppsBlockDMatrix_32f** pMatrix,
   const Ipp32f** mSrc, const int* bSize, int nBlocks);
IppStatus ippsBlockDMatrixInitAlloc_64f(IppsBlockDMatrix_64f** pMatrix,
   const Ipp64f** mSrc, const int* bSize, int nBlocks);
```

BlockDMatrixFree

Deallocates the block diagonal matrix structure.

```
IppStatus ippsBlockDMatrixFree_16s(IppsBlockDMatrix_16s* pMatrix);
IppStatus ippsBlockDMatrixFree_32f(IppsBlockDMatrix_32f* pMatrix);
IppStatus ippsBlockDMatrixFree_64f(IppsBlockDMatrix_64f* pMatrix);
```

NthMaxElement

Searches for the N-th maximal element of a vector.

```
IppStatus ippsNthMaxElement_32s(const Ipp32s* pSrc, int len, int N,
   Ipp32s* pRes);
IppStatus ippsNthMaxElement_32f(const Ipp32f* pSrc, int len, int N,
   Ipp32f* pRes);
IppStatus ippsNthMaxElement_64f(const Ipp64f* pSrc, int len, int N,
   Ipp64f* pRes);
```

VecMatMul

Multiplies a vector by a matrix.

```
IppStatus ippsVecMatMul_16s_D2Sfs(const Ipp16s* pSrc, const Ipp16s*
   pMatr, int step, int height, Ipp16s* pDst, int width, int
   scaleFactor);
IppStatus ippsVecMatMul_16s_D2LSfs(const Ipp16s* pSrc, const Ipp16s**
   mMatr, int height, Ipp16s* pDst, int width, int scaleFactor);
IppStatus ippsVecMatMul_32f_D2(const Ipp32f* pSrc, const Ipp32f* pMatr,
   int step, int height, Ipp32f* pDst, int width);
IppStatus ippsVecMatMul_32f_D2L(const Ipp32f* pSrc, const Ipp32f**
   mMatr, int height, Ipp32f* pDst, int width);
IppStatus ippsVecMatMul_16s32s_D2Sfs(const Ipp16s* pSrc, const Ipp16s*
   pMatr, int step, int height, Ipp32s* pDst, int width, int
   scaleFactor);
IppStatus ippsVecMatMul_16s32s_D2LSfs(const Ipp16s* pSrc, const Ipp16s*
   pMatr, int step, int height, Ipp32s* pDst,int width, int
   scaleFactor);
IppStatus ippsVecMatMul_32s_D2Sfs(const Ipp32s* pSrc, const Ipp32s*
   pMatr, int step, int height, Ipp32s* pDst, int width, int
   scaleFactor);
IppStatus ippsVecMatMul_32s_D2LSfs(const Ipp32s* pSrc, const Ipp32s*
   pMatr, int step, int height, Ipp32s* pDst, int width, int
   scaleFactor);
```

MatVecMul

Multiplies a matrix by a vector.

```
IppStatus ippsMatVecMul_16s_D2Sfs(const Ipp16s* pMatr, int step, const
   Ipp16s* pSrc, int width, Ipp16s* pDst, int height, int scaleFactor);
IppStatus ippsMatVecMul_16s_D2LSfs(const Ipp16s** mMatr, const Ipp16s*
   pSrc, int width, Ipp16s* pDst, int height, int scaleFactor);
```

```
IppStatus ippsMatVecMul_32f_D2(const Ipp32f* pMatr, int step, const
   Ipp32f* pSrc, int width, Ipp32f* pDst, int height);
IppStatus ippsMatVecMul_32f_D2L(const Ipp32f** mMatr, const Ipp32f*
   pSrc, int width, Ipp32f* pDst, int height);
IppStatus ippsMatVecMul_16s32s_D2Sfs(const Ipp16s* pMatr, int step,
   const Ipp16s* pSrc, int width, Ipp32s* pDst, int height, int
   scaleFactor);
IppStatus ippsMatVecMul_16s32s_D2LSfs(const Ipp16s** mMatr, const
   Ipp16s* pSrc, int width, Ipp32s* pDst, int height, int scaleFactor);
IppStatus ippsMatVecMul_32s_D2Sfs(const Ipp32s* pMatr, int step, const
   Ipp32s* pSrc, int width, Ipp32s* pDst, int height, int scaleFactor);
IppStatus ippsMatVecMul_32s_D2LSfs(const Ipp32s** mMatr, const Ipp32s*
   pSrc, int width, Ipp32s* pDst, int height, int scaleFactor);
```

Feature Processing

ZeroMean

Subtracts the mean value from the input vector.

```
IppStatus ippsZeroMean_16s(Ipp16s* pSrcDst, int len);
```

CompensateOffset

Removes the DC offset of the input signals.

```
IppStatus ippsCompensateOffset_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, Ipp16s* pSrcDst0, Ipp16s dst0, Ipp32f val);
IppStatus ippsCompensateOffset_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len, Ipp32f* pSrcDst0, Ipp32f dst0, Ipp32f val);
IppStatus ippsCompensateOffset_16s_I(Ipp16s* pSrcDst, int len, Ipp16s*
   pSrcDst0, Ipp16s dst0, Ipp32f val);
IppStatus ippsCompensateOffset_32f_I(Ipp32f* pSrcDst, int len, Ipp32f*
   pSrcDst0, Ipp32f dst0, Ipp32f val);
IppStatus ippsCompensateOffsetQ15_16s(const Ipp16s* pSrc, Ipp16s* pDst,
   int len, Ipp16s* pSrcDst0, Ipp16s dst0, Ipp16s valQ15);
IppStatus ippsCompensateOffsetQ15_16s_I(Ipp16s* pSrcDst, int len,
   Ipp16s* pSrcDst0, Ipp16s dst0, Ipp16s valQ15);
```

SignChangeRate

Counts the zero-cross rate for the input signal.

```
IppStatus ippsSignChangeRate_16s(const Ipp16s* pSrc, int len, Ipp32s*
   pRes);
IppStatus ippsSignChangeRate_32f(const Ipp32f* pSrc, int len, Ipp32f*
   pRes);
IppStatus ippsSignChangeRate_Count0_16s(const Ipp16s* pSrc, int len,
   Ipp32s* pRes);
```

```
IppStatus ippsSignChangeRate_Count0_32f(const Ipp32f* pSrc, int len,
   Ipp32f* pRes);
IppStatus ippsSignChangeRateXor_32f(const Ipp32f* pSrc, int len, Ipp32f*
   pRes);
```

LinearPrediction

Performs linear prediction analysis on the input vector.

```
IppStatus ippsLinearPrediction_Auto_16s_Sfs(const Ipp16s* pSrc, int
   lenSrc, Ipp16s* pDst, int lenDst, int scaleFactor);
IppStatus ippsLinearPrediction_Auto_32f(const Ipp32f* pSrc, int lenSrc,
   Ipp32f* pDst, int lenDst);
IppStatus ippsLinearPredictionNeg_Auto_16s_Sfs(const Ipp16s* pSrc, int
   lenSrc, Ipp16s* pDst, int lenDst, int scaleFactor);
IppStatus ippsLinearPredictionNeg_Auto_32f(const Ipp32f* pSrc, int
   lenSrc, Ipp32f* pDst, int lenDst);
IppStatus ippsLinearPrediction_Cov_16s_Sfs(const Ipp16s* pSrc, int
   lenSrc, Ipp16s* pDst, int lenDst, int scaleFactor);
IppStatus ippsLinearPrediction_Cov_32f(const Ipp32f* pSrc, int lenSrc,
   Ipp32f* pDst, int lenDst);
```

Durbin

Performs Durbin's recursion on an input vector of autocorrelations.

```
IppStatus ippsDurbin_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   Ipp32f* pErr, int scaleFactor);
IppStatus ippsDurbin_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f* pErr);
```

Schur

Calculates reflection coefficients using Schur algorithm.

```
IppStatus ippsSchur_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int len,
   Ipp32f* pErr, int scaleFactor);
IppStatus ippsSchur_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f* pErr);
```

LPToSpectrum

Calculates smoothed magnitude spectrum.

```
IppStatus ippsLPToSpectrum_16s_Sfs(const Ipp16s* pSrc, int len, Ipp16s*
   pDst, int order, Ipp32s val, int scaleFactor);
IppStatus ippsLPToSpectrum_32f(const Ipp32f* pSrc, int len, Ipp32f* pDst,
   int order, Ipp32f val);
```

LPToCepstrum

Calculates cepstrum coefficients from linear prediction coefficients.

```
IppStatus ippsLPToCepstrum_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, int scaleFactor);
```

```
IppStatus ippsLPToCepstrum_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len);
```

CepstrumToLP

Calculates linear prediction coefficients from cepstrum coefficients.

```
IppStatus ippsCepstrumToLP_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, int scaleFactor);
IppStatus ippsCepstrumToLP_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len);
```

LPToReflection

Calculates the linear prediction reflection coefficients from the linear prediction coefficients.

```
IppStatus ippsLPToReflection_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsLPToReflection_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len);
```

ReflectionToLP

Calculates the linear prediction coefficients from the linear prediction reflection coefficients.

```
IppStatus ippsReflectionToLP_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsReflectionToLP_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len);
```

ReflectionToAR

Converts reflection coefficients to area ratios.

```
IppStatus ippsReflectionToAR_16s_Sfs(const Ipp16s* pSrc, int
   srcShiftVal, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsReflectionToAR_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len);
IppStatus ippsReflectionToLAR_16s_Sfs(const Ipp16s* pSrc, int
   srcShiftVal, Ipp16s* pDst, int len, Ipp32f val, int scaleFactor);
IppStatus ippsReflectionToLAR_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len, Ipp32f val);
IppStatus ippsReflectionToTrueAR_16s_Sfs(const Ipp16s* pSrc, int
   srcShiftVal, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsReflectionToTrueAR_32f(const Ipp32f* pSrc, Ipp32f* pDst,
   int len);
```

ReflectionToTilt

Calculates tilt for rise/fall/connection parameters.

```
IppStatus ippsReflectionToAbsTilt_16s_Sfs(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsReflectionToAbsTilt_32f(const Ipp32f* pSrc1, const Ipp32f*
   pSrc2, Ipp32f* pDst, int len);
```

```
IppStatus ippsReflectionToTilt_16s_Sfs(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsReflectionToTilt_32f(const Ipp32f* pSrc1, const Ipp32f*
   pSrc2, Ipp32f* pDst, int len);
```

PitchmarkToF0

Calculates rise and fall amplitude and duration for tilt.

```
IppStatus ippsPitchmarkToF0Cand_16s_Sfs(const Ipp16s* pSrc, Ipp16s*
   pDst, int len, int scaleFactor);
IppStatus ippsPitchmarkToF0Cand_32f(const Ipp32f* pSrc, Ipp32f* pDst,
   int len);
```

UnitCurve

Calculates tilt for rise and fall coefficicents.

```
IppStatus ippsUnitCurve_16s_Sfs(const Ipp16s* pSrc, int srcShiftVal,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsUnitCurve_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsUnitCurve_16s_ISfs(Ipp16s* pSrcDst, int srcShiftVal, int
   len, int scaleFactor);
IppStatus ippsUnitCurve_32f_I(Ipp32f* pSrcDst, int len);
```

LPToLSP

Calculates line spectrum pairs vector from linear prediction coefficients.

```
IppStatus ippsLPToLSP_16s_Sfs(const Ipp16s* pSrcLP, int srcShiftVal,
   Ipp16s* pDstLSP, int len, int* nRoots, int nInt, int nDiv, int
   scaleFactor);
IppStatus ippsLPToLSP_32f(const Ipp32f* pSrcLP, Ipp32f* pDstLSP, int len,
   int* nRoots, int nInt, int nDiv);
```

LSPToLP

Converts line spectrum pairs vector to linear prediction coefficients.

```
IppStatus ippsLSPToLP_16s_Sfs(const Ipp16s* pSrcLSP, int srcShiftVal,
   Ipp16s* pDstLP, int len, int scaleFactor);
IppStatus ippsLSPToLP_32f(const Ipp32f* pSrcLSP, Ipp32f* pDstLP, int
   len);
```

MelToLinear

Converts Mel-scaled values to linear scale values.

```
IppStatus ippsMelToLinear_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f melMul, Ipp32f melDiv);
```

LinearToMel

Converts linear-scale values to Mel-scale values.

```
IppStatus ippsLinearToMel_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f melMul, Ipp32f melDiv);
```

CopyWithPadding

Copies the input signal to the output with zero-padding.

```
IppStatus ippsCopyWithPadding_16s(const Ipp16s* pSrc, int lenSrc,
   Ipp16s* pDst, int lenDst);
IppStatus ippsCopyWithPadding_32f(const Ipp32f* pSrc, int lenSrc,
   Ipp32f* pDst, int lenDst);
```

MelFBankGetSize

Gets the size of the Mel-frequency filter bank structure.

```
IppStatus ippsMelFBankGetSize_32s(int winSize, int nFilter, IppMelMode
   mode, int* pSize);
```

MelFBankInit

Initializes the structure for performing the Mel-frequency filter bank analysis.

```
IppStatus ippsMelFBankInit_32s(IppsFBankState_32s* pFBank, int* pFFTLen,
   int winSize, Ipp32s sampFreq, Ipp32s lowFreq, Ipp32s highFreq, int
   nFilter, Ipp32s melMulQ15, Ipp32s melDivQ15, IppMelMode mode);
```

MelFBankInitAlloc

Allocates memory and initializes the structure for performing the Mel-frequency filter bank analysis.

```
IppStatus ippsMelFBankInitAlloc_16s(IppsFBankState_16s** pFBank,
   int* pFFTLen, int winSize, Ipp32f sampFreq, Ipp32f lowFreq, Ipp32f
   highFreq, int nFilter, Ipp32f melMul, Ipp32f melDiv, IppMelMode
   mode);
IppStatus ippsMelFBankInitAlloc_32f(IppsFBankState_32f** pFBank,
   int* pFFTLen, int winSize, Ipp32f sampFreq, Ipp32f lowFreq, Ipp32f
   highFreq, int nFilter, Ipp32f melMul, Ipp32f melDiv, IppMelMode
   mode);
```

MelLinFBankInitAlloc

Initializes the structure for performing a combined linear and Mel-frequency filter bank analysis.

```
IppStatus ippsMelLinFBankInitAlloc_16s(IppsFBankState_16s** pFBank,
   int* pFFTLen, int winSize, Ipp32f sampFreq, Ipp32f lowFreq,
   Ipp32f highFreq, int nFilter, Ipp32f highLinFreq, int nLinFilter,
   Ipp32f melMul, Ipp32f melDiv, IppMelMode mode);
IppStatus ippsMelLinFBankInitAlloc_32f(IppsFBankState_32f** pFBank,
   int* pFFTLen, int winSize, Ipp32f sampFreq, Ipp32f lowFreq,
   Ipp32f highFreq, int nFilter, Ipp32f highLinFreq, int nLinFilter,
   Ipp32f melMul, Ipp32f melDiv, IppMelMode mode);
```

EmptyFBankInitAlloc

Initializes an empty filter bank structure.

```
IppStatus ippsEmptyFBankInitAlloc_16s(IppsFBankState_16s** pFBank, int*
   pFFTLen, int winSize, int nFilter, IppMelMode mode);
```

```
IppStatus ippsEmptyFBankInitAlloc_32f(IppsFBankState_32f** pFBank, int*
   pFFTLen, int winSize, int nFilter, IppMelMode mode);
```

FBankFree

Destroys the structure for the filter bank analysis.

```
IppStatus ippsFBankFree_16s(IppsFBankState_16s* pFBank);
IppStatus ippsFBankFree_32f(IppsFBankState_32f* pFBank);
```

FBankGetCenters

Retrieves the center frequencies of the triangular filter banks.

```
IppStatus ippsFBankGetCenters_16s(const IppsFBankState_16s* pFBank, int*
   pCenters);
IppStatus ippsFBankGetCenters_32f(const IppsFBankState_32f* pFBank, int*
   pCenters);
```

FBankSetCenters

Sets the center frequencies of the triangular filter banks.

```
IppStatus ippsFBankSetCenters_16s(IppsFBankState_16s* pFBank, const int*
   pCenters);
IppStatus ippsFBankSetCenters_32f(IppsFBankState_32f* pFBank, const int*
   pCenters);
```

FBankGetCoeffs

Retrieves the filter bank weight coefficients.

```
IppStatus ippsFBankGetCoeffs_16s(const IppsFBankState_16s* pFBank, int
   fIdx, Ipp32f* pCoeffs);
IppStatus ippsFBankGetCoeffs_32f(const IppsFBankState_32f* pFBank, int
   fIdx, Ipp32f* pCoeffs);
```

FBankSetCoeffs

Sets the filter bank weight coefficients.

```
IppStatus ippsFBankSetCoeffs_16s(IppsFBankState_16s* pFBank, int fIdx,
   const Ipp32f* pCoeffs);
IppStatus ippsFBankSetCoeffs_32f(IppsFBankState_32f* pFBank, int fIdx,
   const Ipp32f* pCoeffs);
```

EvalFBank

Performs the filter bank analysis.

```
IppStatus ippsEvalFBank_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, const
   IppsFBankState_16s* pFBank, int scaleFactor);
IppStatus ippsEvalFBank_16s32s_Sfs(const Ipp16s* pSrc, Ipp32s* pDst, const
   IppsFBankState_16s* pFBank, int scaleFactor);
IppStatus ippsEvalFBank_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pDst, const
   IppsFBankState_32s* pFBank, int scaleFactor);
IppStatus ippsEvalFBank_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsFBankState_32f* pFBank);
```

DCTLifterGetSize_MulC0

Gets the size of the DCT structure.

```
IppStatus ippsDCTLifterGetSize_MulC0_16s(int lenDCT, int lenCeps, int*
   pSize);
```

DCTLifterInit_MulC0

Initializes the structure to perform DCT and lift the DCT coefficients.

```
IppStatus ippsDCTLifterInit_MulC0_16s(IppsDCTLifterState_16s*
   pDCTLifter, int lenDCT, const Ipp32s* pLifterQ15, int lenCeps);
```

DCTLifterInitAlloc

Initializes the structure and allocates memory to perform DCT and lift the DCT coefficients.

```
IppStatus ippsDCTLifterInitAlloc_16s(IppsDCTLifterState_16s**
   pDCTLifter, int lenDCT, int lenCeps, int nLifter, Ipp32f val);

IppStatus ippsDCTLifterInitAlloc_C0_16s(IppsDCTLifterState_16s**
   pDCTLifter, int lenDCT, int lenCeps, int nLifter, Ipp32f val, Ipp32f
   val0);

IppStatus ippsDCTLifterInitAlloc_Mul_16s(IppsDCTLifterState_16s**
   pDCTLifter, int lenDCT, const Ipp32f* pLifter, int lenCeps);

IppStatus ippsDCTLifterInitAlloc_MulC0_16s(IppsDCTLifterState_16s**
   pDCTLifter, int lenDCT, const Ipp32f* pLifter, int lenCeps);

IppStatus ippsDCTLifterInitAlloc_32f(IppsDCTLifterState_32f**
   pDCTLifter, int lenDCT, int lenCeps, int nLifter, Ipp32f val);

IppStatus ippsDCTLifterInitAlloc_C0_32f(IppsDCTLifterState_32f**
   pDCTLifter, int lenDCT, int lenCeps, int nLifter, Ipp32f val, Ipp32f
   val0);

IppStatus ippsDCTLifterInitAlloc_Mul_32f(IppsDCTLifterState_32f**
   pDCTLifter, int lenDCT, const Ipp32f* pLifter, int lenCeps);

IppStatus ippsDCTLifterInitAlloc_MulC0_32f(IppsDCTLifterState_32f**
   pDCTLifter, int lenDCT, const Ipp32f* pLifter, int lenCeps);
```

DCTLifterFree

Destroys the structure used for the DCT and liftering.

```
IppStatus ippsDCTLifterFree_16s(IppsDCTLifterState_16s* pDCTLifter);

IppStatus ippsDCTLifterFree_32f(IppsDCTLifterState_32f* pDCTLifter);
```

DCTLifter

Performs the DCT and lifts the DCT coefficients.

```
IppStatus ippsDCTLifter_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, const
   IppsDCTLifterState_16s* pDCTLifter, int scaleFactor);

IppStatus ippsDCTLifter_32s16s_Sfs (const Ipp32s* pSrc, Ipp16s* pDst,
   const IppsDCTLifterState_16s* pDCTLifter, int scaleFactor);

IppStatus ippsDCTLifter_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsDCTLifterState_32f* pDCTLifter);
```

NormEnergy

Normalizes a vector of energy values.

```
IppStatus ippsNormEnergy_32f(Ipp32f* pSrcDst, int step, int height,
   Ipp32f silFloor, Ipp32f enScale);

IppStatus ippsNormEnergy_16s(Ipp16s* pSrcDst, int step, int height,
   Ipp16s silFloor, Ipp16s val, Ipp32f enScale);

IppStatus ippsNormEnergy_RT_32f(Ipp32f* pSrcDst, int step, int height,
   Ipp32f silFloor, Ipp32f maxE, Ipp32f enScale);

IppStatus ippsNormEnergy_RT_16s(Ipp16s* pSrcDst, int step, int height,
   Ipp16s silFloor, Ipp16s maxE, Ipp16s val, Ipp32f enScale);
```

SumMeanVar

Calculates both the sum of a the vector and its square sum.

```
IppStatus ippsSumMeanVar_32f(const Ipp32f* pSrc, int srcStep, int height,
   Ipp32f* pDstMean, Ipp32f* pDstVar, int width);

IppStatus ippsSumMeanVar_32f_I(const Ipp32f* pSrc, int srcStep, int
   height, Ipp32f* pSrcDstMean, Ipp32f* pSrcDstVar, int width);

IppStatus ippsSumMeanVar_16s32f(const Ipp16s* pSrc, int srcStep, int
   height, Ipp32f* pDstMean, Ipp32f* pDstVar, int width);

IppStatus ippsSumMeanVar_16s32f_I(const Ipp16s* pSrc, int srcStep, int
   height, Ipp32f* pSrcDstMean, Ipp32f* pSrcDstVar, int width);

IppStatus ippsSumMeanVar_16s32s_Sfs(const Ipp16s* pSrc, int srcStep, int
   height, Ipp32s* pDstMean, Ipp32s* pDstVar, int width, int
   scaleFactor);

IppStatus ippsSumMeanVar_16s32s_ISfs(const Ipp16s* pSrc, int srcStep,
   int height, Ipp32s* pSrcDstMean, Ipp32s* pSrcDstVar, int width, int
   scaleFactor);
```

NewVar

Calculates the variances given the sum and square sum accumulators.

```
IppStatus ippsNewVar_32f(const Ipp32f* pSrcMean, const Ipp32f* pSrcVar,
   Ipp32f* pDstVar, int width, Ipp32f val1, Ipp32f val2);

IppStatus ippsNewVar_32f_I(const Ipp32f* pSrcMean, Ipp32f* pSrcDstVar,
   int width, Ipp32f val1, Ipp32f val2);

IppStatus ippsNewVar_32s_Sfs(const Ipp32s* pSrcMean, const Ipp32s*
   pSrcVar, Ipp32s* pDstVar, int width, Ipp32f val1, Ipp32f val2, int
   scaleFactor);

IppStatus ippsNewVar_32s_ISfs(const Ipp32s* pSrcMean, Ipp32s*
   pSrcDstVar,
   int width, Ipp32f val1, Ipp32f val2, int scaleFactor);
```

RecSqrt

Calculates square roots of a vector and their reciprocals.

```
IppStatus ippsRecSqrt_32s_Sfs(Ipp32s* pSrcDst, int len, Ipp32s val, int
   scaleFactor);
```

```
IppStatus ippsRecSqrt_32f(Ipp32f* pSrcDst, int len, Ipp32f val);
IppStatus ippsRecSqrt_32s16s_Sfs(const Ipp32s* pSrc, Ipp16s* pDst, int
   len, Ipp32s val, int scaleFactor);
```

AccCovarianceMatrix

Accumulates covariance matrix.

```
IppStatus ippsAccCovarianceMatrix_16s64f_D2L(const Ipp16s** mSrc, int
   height, const Ipp16s* pMean, Ipp64f** mSrcDst, int width, Ipp64f
   val);
IppStatus ippsAccCovarianceMatrix_32f64f_D2L(const Ipp32f** mSrc, int
   height, const Ipp32f* pMean, Ipp64f** mSrcDst, int width, Ipp64f
   val);
IppStatus ippsAccCovarianceMatrix_16s64f_D2(const Ipp16s* pSrc, int
   srcStep, int height, const Ipp16s* pMean, Ipp64f* pSrcDst, int width,
   int dstStep, Ipp64f val);
IppStatus ippsAccCovarianceMatrix_32f64f_D2(const Ipp32f* pSrc, int
   srcStep, int height, const Ipp32f* pMean, Ipp64f* pSrcDst, int width,
   int dstStep, Ipp64f val);
```

CopyColumn

Copies the input sequence into the output sequence.

```
IppStatus ippsCopyColumn_16s_D2(const Ipp16s* pSrc, int srcWidth,
   Ipp16s* pDst, int dstWidth, int height);
IppStatus ippsCopyColumn_32f_D2(const Ipp32f* pSrc, int srcWidth,
   Ipp32f* pDst, int dstWidth, int height);
```

EvalDelta

Calculates the derivatives of feature vectors.

```
IppStatus ippsEvalDelta_16s_D2Sfs(Ipp16s* pSrcDst, int height, int step,
   int width, int offset, int winSize, Ipp16s val, int scaleFactor);
IppStatus ippsEvalDelta_32f_D2(Ipp32f* pSrcDst, int height, int step, int
   width, int offset, int winSize, Ipp32f val);
IppStatus ippsEvalDeltaMul_16s_D2Sfs(Ipp16s* pSrcDst, int height, int
   step, const Ipp16s* pVal, int width, int offset, int winSize, int
   scaleFactor);
IppStatus ippsEvalDeltaMul_32f_D2(Ipp32f* pSrcDst, int height, int step,
   const Ipp32f* pVal, int width, int offset, int winSize);
```

Delta

Copies the base features and calculates the derivatives of feature vectors.

```
IppStatus ippsDelta_Win1_16s_D2Sfs(const Ipp16s* pSrc, int srcWidth,
   Ipp16s* pDst, int dstStep, int height, Ipp16s val, int deltaMode, int
   scaleFactor);
IppStatus ippsDelta_Win2_16s_D2Sfs(const Ipp16s* pSrc, int srcWidth,
   Ipp16s* pDst, int dstStep, int height, Ipp16s val, int deltaMode, int
   scaleFactor);
```

```
IppStatus ippsDelta_Win1_32f_D2(const Ipp32f* pSrc, int srcWidth,
   Ipp32f* pDst, int dstStep, int height, Ipp32f val, int deltaMode);
IppStatus ippsDelta_Win2_32f_D2(const Ipp32f* pSrc, int srcWidth,
   Ipp32f* pDst, int dstStep, int height, Ipp32f val, int deltaMode);
IppStatus ippsDeltaMul_Win1_16s_D2Sfs(const Ipp16s* pSrc, const Ipp16s*
   pVal, int srcWidth, Ipp16s* pDst, int dstStep, int height, int
   deltaMode, int scaleFactor);
IppStatus ippsDeltaMul_Win2_16s_D2Sfs(const Ipp16s* pSrc, const Ipp16s*
   pVal, int srcWidth, Ipp16s* pDst, int dstStep, int height, int
   deltaMode, int scaleFactor);
IppStatus ippsDeltaMul_Win1_32f_D2(const Ipp32f* pSrc, const Ipp32f*
   pVal, int srcWidth, Ipp32f* pDst, int dstStep, int height, int
   deltaMode);
IppStatus ippsDeltaMul_Win2_32f_D2(const Ipp32f* pSrc, const Ipp32f*
   pVal, int srcWidth, Ipp32f* pDst, int dstStep, int height, int
   deltaMode);
```

DeltaDelta

Copies the base features and calculates their first and second derivatives.

```
IppStatus ippsDeltaDelta_Win1_16s_D2Sfs(const Ipp16s* pSrc, int
   srcWidth, Ipp16s* pDst, int dstStep, int height, Ipp16s val1, Ipp16s
   val2, int deltaMode, int scaleFactor);
IppStatus ippsDeltaDelta_Win2_16s_D2Sfs(const Ipp16s* pSrc, int
   srcWidth, Ipp16s* pDst, int dstStep, int height, Ipp16s val1, Ipp16s
   val2, int deltaMode, int scaleFactor);
IppStatus ippsDeltaDelta_Win1_32f_D2(const Ipp32f* pSrc, int srcWidth,
   Ipp32f* pDst, int dstStep, int height, Ipp32f val1, Ipp32f val2, int
   deltaMode);
IppStatus ippsDeltaDelta_Win2_32f_D2(const Ipp32f* pSrc, int srcWidth,
   Ipp32f* pDst, int dstStep, int height, Ipp32f val1, Ipp32f val2, int
   deltaMode);
IppStatus ippsDeltaDeltaMul_Win1_16s_D2Sfs(const Ipp16s* pSrc, const
   Ipp16s* pVal, int srcWidth, Ipp16s* pDst, int dstStep, int height,
   int deltaMode, int scaleFactor);
IppStatus ippsDeltaDeltaMul_Win2_16s_D2Sfs(const Ipp16s* pSrc, const
   Ipp16s* pVal, int srcWidth, Ipp16s* pDst, int dstStep, int height,
   int deltaMode, int scaleFactor);
IppStatus ippsDeltaDeltaMul_Win1_32f_D2(const Ipp32f* pSrc, const
   Ipp32f* pVal, int srcWidth, Ipp32f* pDst, int dstStep, int height,
   int deltaMode);
IppStatus ippsDeltaDeltaMul_Win2_32f_D2(const Ipp32f* pSrc, const
   Ipp32f* pVal, int srcWidth, Ipp32f* pDst, int dstStep, int height,
   int deltaMode);
```

CrossCorrCoeffDecim
Calculates vector of cross correlation coefficients with decimation.

```
IppStatus ippsCrossCorrCoeffDecim_16s32f(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, int maxLen, int minLen, Ipp32f* pDst, int dec);
```

CrossCorrCoeff
Calculates the cross correlation coefficient.

```
IppStatus ippsCrossCorrCoeff_16s32f(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32f* pResult);
IppStatus ippsCrossCorrCoeffPartial_16s32f(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, int len, Ipp32f val, Ipp32f* pResult);
```

CrossCorrCoeffInterpolation
Calculates interpolated cross correlation coefficient.

```
IppStatus ippsCrossCorrCoeffInterpolation_16s32f(const Ipp16s* pSrc1,
   const Ipp16s* pSrc2, int len, Ipp32f* pBeta, Ipp32f* pResult);
```

Model Evaluation

AddNRows
Adds N vectors from a vector array.

```
IppStatus ippsAddNRows_32f_D2(Ipp32f* pSrc, int height, int offset,
   int step, Ipp32s* pInd, Ipp16u* pAddInd, int rows, Ipp32f* pDst,
   int width, Ipp32f weight);
```

ScaleLM
Scales vector elements with thresholding.

```
IppStatus ippsScaleLM_16s32s(const Ipp16s* pSrc, Ipp32s* pDst, int len,
   Ipp16s floor, Ipp16s scale, Ipp32s base);
IppStatus ippsScaleLM_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len,
   Ipp32f floor, Ipp32f scale, Ipp32f base);
```

LogAdd
Adds two vectors in the logarithmic representation.

```
IppStatus ippsLogAdd_32f(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len,
   IppHintAlgorithm hint);
IppStatus ippsLogAdd_64f(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len,
   IppHintAlgorithm hint);
IppStatus ippsLogAdd_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   len, int scaleFactor, IppHintAlgorithm hint);
IppStatus ippsLogAdd_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int
   len, int scaleFactor, IppHintAlgorithm hint);
```

LogSub

Subtracts a vector from another vector, in the logarithmic representation.

```
IppStatus ippsLogSub_32f(const Ipp32f* pSrc, Ipp32f* pSrcDst, int len);
IppStatus ippsLogSub_64f(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len);
IppStatus ippsLogSub_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pSrcDst, int
    len, int scaleFactor);
IppStatus ippsLogSub_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pSrcDst, int
    len, int scaleFactor);
```

LogSum

Sums vector elements in the logarithmic representation.

```
IppStatus ippsLogSum_32f(const Ipp32f* pSrc, Ipp32f* pResult, int len,
    IppHintAlgorithm hint);
IppStatus ippsLogSum_64f(const Ipp64f* pSrc, Ipp64f* pResult, int len,
    IppHintAlgorithm hint);
IppStatus ippsLogSum_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pResult, int
    len, int scaleFactor, IppHintAlgorithm hint);
IppStatus ippsLogSum_32s_Sfs(const Ipp32s* pSrc, Ipp32s* pResult, int
    len, int scaleFactor, IppHintAlgorithm hint);
```

MahDistSingle

Calculates the Mahalanobis distance
for a single observation vector.

```
IppStatus ippsMahDistSingle_16s32s_Sfs(const Ipp16s* pSrc, const Ipp16s*
    pMean, const Ipp16s* pVar, int len, Ipp32s* pResult, int
    scaleFactor);
IppStatus ippsMahDistSingle_16s32f(const Ipp16s* pSrc, const Ipp16s*
    pMean, const Ipp16s* pVar, int len, Ipp32f* pResult);
IppStatus ippsMahDistSingle_32f(const Ipp32f* pSrc, const Ipp32f* pMean,
    const Ipp32f* pVar, int len, Ipp32f* pResult);
IppStatus ippsMahDistSingle_64f(const Ipp64f* pSrc, const Ipp64f* pMean,
    const Ipp64f* pVar, int len, Ipp64f* pResult);
IppStatus ippsMahDistSingle_32f64f(const Ipp32f* pSrc, const Ipp32f*
    pMean, const Ipp32f* pVar, int len, Ipp64f* pResult);
```

MahDist

Calculates the Mahalanobis distances for multiple observation vectors.

```
IppStatus ippsMahDist_32f_D2(const Ipp32f* pSrc, int step, const Ipp32f*
    pMean, const Ipp32f* pVar, int width, Ipp32f* pDst, int height);
IppStatus ippsMahDist_32f_D2L(const Ipp32f** mSrc, const Ipp32f* pMean,
    const Ipp32f* pVar, int width, Ipp32f* pDst, int height);
IppStatus ippsMahDist_64f_D2(const Ipp64f* pSrc, int step, const Ipp64f*
    pMean, const Ipp64f* pVar, int width, Ipp64f* pDst, int height);
```

```
IppStatus ippsMahDist_64f_D2L(const Ipp64f** mSrc, const Ipp64f* pMean,
   const Ipp64f* pVar, int width, Ipp64f* pDst, int height);
```

MahDistMultiMix

Calculates the Mahalanobis distances for multiple means and variances.

```
IppStatus ippsMahDistMultiMix_32f_D2(const Ipp32f* pMean, const Ipp32f*
   pVar, int step, const Ipp32f* pSrc, int width, Ipp32f* pDst, int
   height);

IppStatus ippsMahDistMultiMix_32f_D2L(const Ipp32f** mMean, const
   Ipp32f** mVar, const Ipp32f* pSrc, int width, Ipp32f* pDst, int
   height);

IppStatus ippsMahDistMultiMix_64f_D2(const Ipp64f* pMean, const Ipp64f*
   pVar, int step, const Ipp64f* pSrc, int width, Ipp64f* pDst, int
   height);

IppStatus ippsMahDistMultiMix_64f_D2L(const Ipp64f** mMean, const
   Ipp64f** mVar, const Ipp64f* pSrc, int width, Ipp64f* pDst, int
   height);
```

LogGaussSingle

Calculates the observation probability for a single Gaussian with an observation vector.

```
IppStatus ippsLogGaussSingle_16s32s_Sfs(const Ipp16s* pSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32s* pResult, Ipp32s
   val, int scaleFactor);

IppStatus ippsLogGaussSingle_Scaled_16s32f(const Ipp16s* pSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32f* pResult, Ipp32f
   val, int scaleFactor);

IppStatus ippsLogGaussSingle_32f(const Ipp32f* pSrc, const Ipp32f*
   pMean, const Ipp32f* pVar, int len, Ipp32f* pResult, Ipp32f val);

IppStatus ippsLogGaussSingle_32f64f(const Ipp32f* pSrc, const Ipp32f*
   pMean, const Ipp32f* pVar, int len, Ipp64f* pResult, Ipp64f val);

IppStatus ippsLogGaussSingle_64f(const Ipp64f* pSrc, const Ipp64f*
   pMean, const Ipp64f* pVar, int len, Ipp64f* pResult, Ipp64f val);

IppStatus ippsLogGaussSingle_Low_16s32s_Sfs(const Ipp16s* pSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32s* pResult, Ipp32s
   val, int scaleFactor);

IppStatus ippsLogGaussSingle_LowScaled_16s32f(const Ipp16s* pSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32f* pResult, Ipp32f
   val, int scaleFactor);

IppStatus ippsLogGaussSingle_DirectVar_16s32s_Sfs(const Ipp16s* pSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32s* pResult,
   Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussSingle_DirectVarScaled_16s32f(const Ipp16s* pSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int len, Ipp32f* pResult,
   Ipp32f val, int scaleFactor);
```

```
IppStatus ippsLogGaussSingle_DirectVar_32f(const Ipp32f* pSrc, const
   Ipp32f* pMean, const Ipp32f* pVar, int len, Ipp32f* pResult, Ipp32f
   val);
IppStatus ippsLogGaussSingle_DirectVar_32f64f(const Ipp32f* pSrc, const
   Ipp32f* pMean, const Ipp32f* pVar, int len, Ipp64f* pResult, Ipp64f
   val);
IppStatus ippsLogGaussSingle_DirectVar_64f(const Ipp64f* pSrc, const
   Ipp64f* pMean, const Ipp64f* pVar, int len, Ipp64f* pResult, Ipp64f
   val);
IppStatus ippsLogGaussSingle_IdVar_16s32s_Sfs(const Ipp16s* pSrc, const
   Ipp16s* pMean, int len, Ipp32s* pResult, Ipp32s val, int
   scaleFactor);
IppStatus ippsLogGaussSingle_IdVarScaled_16s32f(const Ipp16s* pSrc,
   const Ipp16s* pMean, int len, Ipp32f* pResult, Ipp32f val, int
   scaleFactor);
IppStatus ippsLogGaussSingle_IdVar_32f(const Ipp32f* pSrc, const Ipp32f*
   pMean, int len, Ipp32f* pResult, Ipp32f val);
IppStatus ippsLogGaussSingle_IdVar_32f64f(const Ipp32f* pSrc, const
   Ipp32f* pMean, int len, Ipp64f* pResult, Ipp64f val);
IppStatus ippsLogGaussSingle_IdVar_64f(const Ipp64f* pSrc, const Ipp64f*
   pMean, int len, Ipp64f* pResult, Ipp64f val);
IppStatus ippsLogGaussSingle_IdVarLow_16s32s_Sfs(const Ipp16s* pSrc,
   const Ipp16s* pMean, int len, Ipp32s* pResult, Ipp32s val, int
   scaleFactor);
IppStatus ippsLogGaussSingle_IdVarLowScaled_16s32f(const Ipp16s* pSrc,
   const Ipp16s* pMean, int len, Ipp32f* pResult, Ipp32f val, int
   scaleFactor);
IppStatus ippsLogGaussSingle_BlockDVar_16s32s_Sfs(const Ipp16s* pSrc,
   const Ipp16s* pMean, const IppsBlockDMatrix_16s*  pBlockVar, int len,
   Ipp32s* pResult, Ipp32s val, int scaleFactor);
IppStatus ippsLogGaussSingle_BlockDVarScaled_16s32f(const Ipp16s* pSrc,
   const Ipp16s* pMean, const IppsBlockDMatrix_16s* pBlockVar, int len,
   Ipp32f* pResult, Ipp32f val, int scaleFactor);
IppStatus ippsLogGaussSingle_BlockDVar_32f(const Ipp32f* pSrc, const
   Ipp32f* pMean, const IppsBlockDMatrix_32f* pBlockVar, int len,
   Ipp32f* pResult, Ipp32f val);
IppStatus ippsLogGaussSingle_BlockDVar_32f64f(const Ipp32f* pSrc, const
   Ipp32f* pMean, const IppsBlockDMatrix_32f*  pBlockVar, int len,
   Ipp64f* pResult, Ipp64f val);
IppStatus ippsLogGaussSingle_BlockDVar_64f(const Ipp64f* pSrc, const
   Ipp64f* pMean, const IppsBlockDMatrix_64f*  pBlockVar, int len,
   Ipp64f* pResult, Ipp64f val);
```

LogGauss

Calculates the observation probability for a single Gaussian with multiple observation vectors.

```
IppStatus ippsLogGauss_16s32s_D2Sfs(const Ipp16s* pSrc, int step, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pDst, int
   height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGauss_16s32s_D2LSfs(const Ipp16s** mSrc, const Ipp16s*
   pMean, const Ipp16s* pVar, int width, Ipp32s* pDst, int height,
   Ipp32s val, int scaleFactor);

IppStatus ippsLogGauss_Scaled_16s32f_D2(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_Scaled_16s32f_D2L(const Ipp16s** mSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_32f_D2(const Ipp32f* pSrc, int step, const Ipp32f*
   pMean, const Ipp32f* pVar, int width, Ipp32f* pDst, int height,
   Ipp32f val);

IppStatus ippsLogGauss_32f_D2L(const Ipp32f** mSrc, const Ipp32f* pMean,
   const Ipp32f* pVar, int width, Ipp32f* pDst, int height, Ipp32f val);

IppStatus ippsLogGauss_64f_D2(const Ipp64f* pSrc, int step, const Ipp64f*
   pMean, const Ipp64f* pVar, int width, Ipp64f* pDst, int height,
   Ipp64f val);

IppStatus ippsLogGauss_64f_D2L(const Ipp64f** mSrc, const Ipp64f* pMean,
   const Ipp64f* pVar, int width, Ipp64f* pDst, int height, Ipp64f val);

IppStatus ippsLogGauss_Low_16s32s_D2Sfs(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pDst, int
   height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGauss_Low_16s32s_D2LSfs(const Ipp16s** mSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pDst, int
   height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGauss_LowScaled_16s32f_D2(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_LowScaled_16s32f_D2L(const Ipp16s** mSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_IdVar_16s32s_D2Sfs(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, int width, Ipp32s* pDst, int height, Ipp32s val,
   int scaleFactor);

IppStatus ippsLogGauss_IdVar_16s32s_D2LSfs(const Ipp16s** mSrc, const
   Ipp16s* pMean, int width, Ipp32s* pDst, int height, Ipp32s val, int
   scaleFactor);
```

```
IppStatus ippsLogGauss_IdVarScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32f* pDst, int height,
    Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_IdVarScaled_16s32f_D2L(const Ipp16s** mSrc, const
    Ipp16s* pMean, int width, Ipp32f* pDst, int height, Ipp32f val,
    int scaleFactor);

IppStatus ippsLogGauss_IdVar_32f_D2(const Ipp32f* pSrc, int step, const
    Ipp32f* pMean, int width, Ipp32f* pDst, int height, Ipp32f val);

IppStatus ippsLogGauss_IdVar_32f_D2L(const Ipp32f** mSrc, const Ipp32f*
    pMean, int width, Ipp32f* pDst, int height, Ipp32f val);

IppStatus ippsLogGauss_IdVar_64f_D2(const Ipp64f* pSrc, int step, const
    Ipp64f* pMean, int width, Ipp64f* pDst, int height, Ipp64f val);

IppStatus ippsLogGauss_IdVar_64f_D2L(const Ipp64f** mSrc, const Ipp64f*
    pMean, int width, Ipp64f* pDst, int height, Ipp64f val);

IppStatus ippsLogGauss_IdVarLow_16s32s_D2Sfs(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32s* pDst, int height,
    Ipp32s val, int scaleFactor);

IppStatus ippsLogGauss_IdVarLow_16s32s_D2LSfs(const Ipp16s** mSrc, const
    Ipp16s* pMean, int width, Ipp32s* pDst, int height, Ipp32s val, int
    scaleFactor);

IppStatus ippsLogGauss_IdVarLowScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32f* pDst, int height,
    Ipp32f val, int scaleFactor);

IppStatus ippsLogGauss_IdVarLowScaled_16s32f_D2L(const Ipp16s** mSrc,
    const Ipp16s* pMean, int width, Ipp32f* pDst, int height, Ipp32f val,
    int scaleFactor);
```

LogGaussMultiMix

Calculates the observation probability for multiple Gaussian mixture components.

```
IppStatus ippsLogGaussMultiMix_16s32s_D2Sfs(const Ipp16s* pMean, const
    Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, Ipp32s*
    pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_16s32s_D2LSfs(const Ipp16s** mMean, const
    Ipp16s** mVar, const Ipp16s* pSrc, int width, Ipp32s* pSrcDst, int
    height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_Scaled_16s32f_D2(const Ipp16s* pMean,
    const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, Ipp32f*
    pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_Scaled_16s32f_D2L(const Ipp16s** mMean,
    const Ipp16s** mVar, const Ipp16s* pSrc, int width, Ipp32f* pSrcDst,
    int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_32f_D2(const Ipp32f* pMean, const Ipp32f*
    pVar, int step, const Ipp32f* pSrc, int width, Ipp32f* pSrcDst, int
    height);
```

```
IppStatus ippsLogGaussMultiMix_32f_D2L(const Ipp32f** mMean, const
   Ipp32f** mVar, const Ipp32f* pSrc, int width, Ipp32f* pSrcDst, int
   height);

IppStatus ippsLogGaussMultiMix_64f_D2(const Ipp64f* pMean, const Ipp64f*
   pVar, int step, const Ipp64f* pSrc, int width, Ipp64f* pSrcDst, int
   height);

IppStatus ippsLogGaussMultiMix_64f_D2L(const Ipp64f** mMean, const
   Ipp64f** mVar, const Ipp64f* pSrc, int width, Ipp64f* pSrcDst, int
   height);

IppStatus ippsLogGaussMultiMix_Low_16s32s_D2Sfs(const Ipp16s* pMean,
   const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, Ipp32s*
   pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_Low_16s32s_D2LSfs(const Ipp16s** mMean,
   const Ipp16s** mVar, const Ipp16s* pSrc, int width, Ipp32s* pSrcDst,
   int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_LowScaled_16s32f_D2(const Ipp16s* pSrc,
   int step, const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f*
   pDst, int height, int scaleFactor);

IppStatus ippsLogGaussMultiMix_LowScaled_16s32f_D2L(const Ipp16s** mSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, int scaleFactor);
```

LogGaussMax

Calculates the likelihood probability given multiple observations and a Gaussian mixture component, using the maximum operation.

```
IppStatus ippsLogGaussMax_16s32s_D2Sfs(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pSrcDst,
   int height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_16s32s_D2LSfs(const Ipp16s** mSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pSrcDst, int
   height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_Scaled_16s32f_D2(const Ipp16s* pSrc, int step,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_Scaled_16s32f_D2L(const Ipp16s** mSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst, int
   height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_32f_D2(const Ipp32f* pSrc, int step, const
   Ipp32f* pMean, const Ipp32f* pVar, int width, Ipp32f* pSrcDst, int
   height, Ipp32f val);

IppStatus ippsLogGaussMax_32f_D2L(const Ipp32f** mSrc, const Ipp32f*
   pMean, const Ipp32f* pVar, int width, Ipp32f* pSrcDst, int height,
   Ipp32f val);
```

```
IppStatus ippsLogGaussMax_64f_D2(const Ipp64f* pSrc, int step, const
    Ipp64f* pMean, const Ipp64f* pVar, int width, Ipp64f* pSrcDst, int
    height, Ipp64f val);

IppStatus ippsLogGaussMax_64f_D2L(const Ipp64f** mSrc, const Ipp64f*
    pMean, const Ipp64f* pVar, int width, Ipp64f* pSrcDst, int height,
    Ipp64f val);

IppStatus ippsLogGaussMax_Low_16s32s_D2Sfs(const Ipp16s* pSrc, int step,
    const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pSrcDst,
    int height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_Low_16s32s_D2LSfs(const Ipp16s** mSrc, const
    Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pSrcDst, int
    height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_LowScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f*
    pSrcDst, int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_LowScaled_16s32f_D2L(const Ipp16s** mSrc,
    const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst,
    int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVar_16s32s_D2Sfs(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32s* pSrcDst, int height,
    Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVar_16s32s_D2LSfs(const Ipp16s** mSrc, const
    Ipp16s* pMean, int width, Ipp32s* pSrcDst, int height, Ipp32s val,
    int scaleFactor);

IppStatus ippsLogGaussMax_IdVarScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int height,
    Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVarScaled_16s32f_D2L(const Ipp16s** mSrc,
    const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f
    val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVar_32f_D2(const Ipp32f* pSrc, int step,
    const Ipp32f* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f
    val);

IppStatus ippsLogGaussMax_IdVar_32f_D2L(const Ipp32f** mSrc, const
    Ipp32f* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f val);

IppStatus ippsLogGaussMax_IdVar_64f_D2(const Ipp64f* pSrc, int step,
    const Ipp64f* pMean, int width, Ipp64f* pSrcDst, int height, Ipp64f
    val);

IppStatus ippsLogGaussMax_IdVar_64f_D2L(const Ipp64f** mSrc, const
    Ipp64f* pMean, int width, Ipp64f* pSrcDst, int height, Ipp64f val);

IppStatus ippsLogGaussMax_IdVarLow_16s32s_D2Sfs(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s*
    pSrcDst, int height, Ipp32s val, int scaleFactor);
```

```
IppStatus ippsLogGaussMax_IdVarLow_16s32s_D2LSfs(const Ipp16s** mSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32s* pSrcDst,
   int height, Ipp32s val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVarLowScaled_16s32f_D2(const Ipp16s* pSrc,
   int step, const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f*
   pSrcDst, int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussMax_IdVarLowScaled_16s32f_D2L(const Ipp16s** mSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst,
   int height, Ipp32f val, int scaleFactor);
```

LogGaussMaxMultiMix

Calculate the likelihood probability for multiple Gaussian mixture components, using the maximum operation.

```
IppStatus ippsLogGaussMaxMultiMix_16s32s_D2Sfs(const Ipp16s* pMean,
   const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, const
   Ipp32s* pVal, Ipp32s* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_16s32s_D2LSfs(const Ipp16s** mMean,
   const Ipp16s** mVar, const Ipp16s* pSrc, int width, const Ipp32s*
   pVal, Ipp32s* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_Scaled_16s32f_D2(const Ipp16s* pMean,
   const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, const
   Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_Scaled_16s32f_D2L(const Ipp16s**
   mMean, const Ipp16s** mVar, const Ipp16s* pSrc, int width, const
   Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_32f_D2(const Ipp32f* pMean, const
   Ipp32f* pVar, int step, const Ipp32f* pSrc, int width, const Ipp32f*
   pVal, Ipp32f* pSrcDst, int height);

IppStatus ippsLogGaussMaxMultiMix_32f_D2L(const Ipp32f** mMean, const
   Ipp32f** mVar, const Ipp32f* pSrc, int width, const Ipp32f* pVal,
   Ipp32f* pSrcDst, int height);

IppStatus ippsLogGaussMaxMultiMix_64f_D2(const Ipp64f* pMean, const
   Ipp64f* pVar, int step, const Ipp64f* pSrc, int width, const Ipp64f*
   pVal, Ipp64f* pSrcDst, int height);

IppStatus ippsLogGaussMaxMultiMix_64f_D2L(const Ipp64f** mMean, const
   Ipp64f** mVar, const Ipp64f* pSrc, int width, const Ipp64f* pVal,
   Ipp64f* pSrcDst, int height);

IppStatus ippsLogGaussMaxMultiMix_Low_16s32s_D2Sfs(const Ipp16s* pMean,
   const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, const
   Ipp32s* pVal, Ipp32s* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_Low_16s32s_D2LSfs(const Ipp16s**
   mMean, const Ipp16s** mVar, const Ipp16s* pSrc, int width, const
   Ipp32s* pVal, Ipp32s* pSrcDst, int height, int scaleFactor);
```

```
IppStatus ippsLogGaussMaxMultiMix_LowScaled_16s32f_D2(const Ipp16s*
    pMean, const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width,
    const Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);

IppStatus ippsLogGaussMaxMultiMix_LowScaled_16s32f_D2L(const Ipp16s**
    mMean, const Ipp16s** mVar, const Ipp16s* pSrc, int width, const
    Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);
```

LogGaussAdd

Calculates the likelihood probability for multiple observation vectors.

```
IppStatus ippsLogGaussAdd_Scaled_16s32f_D2(const Ipp16s* pSrc, int step,
    const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst,
    int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussAdd_Scaled_16s32f_D2L(const Ipp16s** mSrc, const
    Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst, int
    height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussAdd_32f_D2(const Ipp32f** pSrc, int step, const
    Ipp32f* pMean, const Ipp32f* pVar, int width, Ipp32f* pSrcDst, int
    height, Ipp32f val);

IppStatus ippsLogGaussAdd_32f_D2L(const Ipp32f** mSrc, const Ipp32f*
    pMean, const Ipp32f* pVar, int width, Ipp32f* pSrcDst, int height,
    Ipp32f val);

IppStatus ippsLogGaussAdd_64f_D2(const Ipp64f* pSrc, int step, const
    Ipp64f* pMean, const Ipp64f* pVar, int width, Ipp64f* pSrcDst, int
    height, Ipp64f val);

IppStatus ippsLogGaussAdd_64f_D2L(const Ipp64f** mSrc, const Ipp64f*
    pMean, const Ipp64f* pVar, int width, Ipp64f* pSrcDst, int height,
    Ipp64f val);

IppStatus ippsLogGaussAdd_LowScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f*
    pSrcDst, int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussAdd_LowScaled_16s32f_D2L(const Ipp16s** mSrc,
    const Ipp16s* pMean, const Ipp16s* pVar, int width, Ipp32f* pSrcDst,
    int height, Ipp32f val, int scaleFactor);

IppStatus ippsLogGaussAdd_IdVarScaled_16s32f_D2(const Ipp16s* pSrc, int
    step, const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int height,
    Ipp32f val,
    int scaleFactor);

IppStatus ippsLogGaussAdd_IdVarScaled_16s32f_D2L(const Ipp16s** mSrc,
    const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f
    val,
    int scaleFactor);

IppStatus ippsLogGaussAdd_IdVar_32f_D2(const Ipp32f* pSrc, int step,
    const Ipp32f* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f
    val);

IppStatus ippsLogGaussAdd_IdVar_32f_D2L(const Ipp32f** mSrc, const
    Ipp32f* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f val);
```

```
IppStatus ippsLogGaussAdd_IdVar_64f_D2(const Ipp64f* pSrc, int step,
   const Ipp64f* pMean, int width, Ipp64f* pSrcDst, int height, Ipp64f
   val);
IppStatus ippsLogGaussAdd_IdVar_64f_D2L(const Ipp64f** mSrc, const
   Ipp64f* pMean, int width, Ipp64f* pSrcDst, int height, Ipp64f val);
IppStatus ippsLogGaussAdd_IdVarLowScaled_16s32f_D2(const Ipp16s* pSrc,
   int step, const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int
   height, Ipp32f val, int scaleFactor);
IppStatus ippsLogGaussAdd_IdVarLowScaled_16s32f_D2L(const Ipp16s** mSrc,
   const Ipp16s* pMean, int width, Ipp32f* pSrcDst, int height, Ipp32f
   val, int scaleFactor);
```

LogGaussAddMultiMix

Calculates the likelihood probability for multiple Gaussian mixture components.

```
IppStatus ippsLogGaussAddMultiMix_Scaled_16s32f_D2(const Ipp16s* pMean,
   const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width, const
   Ipp32f pVal, Ipp32f* pSrcDst, int height, int scaleFactor);
IppStatus ippsLogGaussAddMultiMix_Scaled_16s32f_D2L(const Ipp16s**
   mMean, const Ipp16s** mVar, const Ipp16s* pSrc, int width, const
   Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);
IppStatus ippsLogGaussAddMultiMix_32f_D2(const Ipp32f* pMean, const
   Ipp32f* pVar, int step, const Ipp32f* pSrc, int width, const Ipp32f*
   pVal, Ipp32f* pSrcDst, int height);
IppStatus ippsLogGaussAddMultiMix_32f_D2L(const Ipp32f** mMean, const
   Ipp32f** mVar, const Ipp32f* pSrc, int width, const Ipp32f* pVal,
   Ipp32f* pSrcDst, int height);
IppStatus ippsLogGaussAddMultiMix_64f_D2(const Ipp64f* pMean, const
   Ipp64f* pVar, int step, const Ipp64f* pSrc, int width, const Ipp64f*
   pVal, Ipp64f* pSrcDst, int height);
IppStatus ippsLogGaussAddMultiMix_64f_D2L(const Ipp64f** mMean, const
   Ipp64f** mVar, const Ipp64f* pSrc, int width, const Ipp64f* pVal,
   Ipp64f* pSrcDst, int height);
IppStatus ippsLogGaussAddMultiMix_LowScaled_16s32f_D2(const Ipp16s*
   pMean, const Ipp16s* pVar, int step, const Ipp16s* pSrc, int width,
   const Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);
IppStatus ippsLogGaussAddMultiMix_LowScaled_16s32f_D2L(const Ipp16s**
   mMean, const Ipp16s** mVar, const Ipp16s* pSrc, int width, const
   Ipp32f* pVal, Ipp32f* pSrcDst, int height, int scaleFactor);
```

LogGaussMixture

Calculates the likelihood probability for the Gaussian mixture.

```
IppStatus ippsLogGaussMixture_Scaled_16s32f_D2(const Ipp16s* pSrc, const
   Ipp16s* pMean, const Ipp16s* pVar, int height, int step, int width,
   const Ipp32f* pVal, Ipp32f* pResult, int scaleFactor);
```

```
IppStatus ippsLogGaussMixture_Scaled_16s32f_D2L(const Ipp16s* pSrc,
    const Ipp16s** mMean, const Ipp16s** mVar, int height, int width,
    const Ipp32f* pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_LowScaled_16s32f_D2(const Ipp16s* pSrc,
    const Ipp16s* pMean, const Ipp16s* pVar, int height, int step, int
    width, const Ipp32f* pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_LowScaled_16s32f_D2L(const Ipp16s* pSrc,
    const Ipp16s** mMean, const Ipp16s** mVar, int height, int width,
    const Ipp32f* pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_32f_D2(const Ipp32f* pSrc, const Ipp32f*
    pMean, const Ipp32f* pVar, int height, int step, int width, const
    Ipp32f* pVal, Ipp32f* pResult);

IppStatus ippsLogGaussMixture_32f_D2L(const Ipp32f* pSrc, const Ipp32f**
    mMean, const Ipp32f** mVar, int height, int width, const Ipp32f*
    pVal, Ipp32f* pResult);

IppStatus ippsLogGaussMixture_64f_D2(const Ipp64f* pSrc, const Ipp64f*
    pMean, const Ipp64f* pVar, int height, int step, int width, const
    Ipp64f* pVal, Ipp64f* pResult);

IppStatus ippsLogGaussMixture_64f_D2L(const Ipp64f* pSrc, const Ipp64f**
    mMean, const Ipp64f** mVar, int height, int width, const Ipp64f*
    pVal, Ipp64f* pResult);

IppStatus ippsLogGaussMixture_IdVarScaled_16s32f_D2(const Ipp16s* pSrc,
    const Ipp16s* pMean, int height, int step, int width, const Ipp32f*
    pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_IdVarScaled_16s32f_D2L(const Ipp16s* pSrc,
    const Ipp16s** mMean, int height, int width, const Ipp32f* pVal,
    Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_IdVarLowScaled_16s32f_D2(const Ipp16s*
    pSrc, const Ipp16s* pMean, int height, int step, int width, const
    Ipp32f* pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_IdVarLowScaled_16s32f_D2L(const Ipp16s*
    pSrc, const Ipp16s** mMean, int height, int width, const Ipp32f*
    pVal, Ipp32f* pResult, int scaleFactor);

IppStatus ippsLogGaussMixture_IdVar_32f_D2(const Ipp32f* pSrc, const
    Ipp32f* pMean, int height, int step, int width, const Ipp32f* pVal,
    Ipp32f* pResult);

IppStatus ippsLogGaussMixture_IdVar_32f_D2L(const Ipp32f* pSrc, const
    Ipp32f** mMean, int height, int width, const Ipp32f* pVal, Ipp32f*
    pResult);

IppStatus ippsLogGaussMixture_IdVar_64f_D2(const Ipp64f* pSrc, const
    Ipp64f* pMean,  int height, int step, int width, const Ipp64f* pVal,
    Ipp64f* pResult);

IppStatus ippsLogGaussMixture_IdVar_64f_D2L(const Ipp64f* pSrc, const
    Ipp64f** mMean, int height, int width, const Ipp64f* pVal, Ipp64f*
    pResult);
```

LogGaussMixtureSelect

Calculates the likelihood probability for the Gaussian mixture using Gaussian selection.

```
IppStatus ippsLogGaussMixture_SelectScaled_16s32f_D2(const Ipp16s* pSrc,
   const Ipp16s* pMean, const Ipp16s* pVar, int step, int width, const
   Ipp32s* pVal, const Ipp8u* pSign, int height, Ipp32s* pResult, int
   frames, Ipp32f none, int scaleFactor);

IppStatus ippsLogGaussMixture_SelectScaled_16s32f_D2L(const Ipp16s**
   mSrc, const Ipp16s** mMean, const Ipp16s** mVar, int width, const
   Ipp32s* pVal, const Ipp8u* pSign, int height, Ipp32s* pResult, int
   frames, Ipp32f none, int scaleFactor);

IppStatus ippsLogGaussMixture_Select_32f_D2(const Ipp32f* pSrc, const
   Ipp32f* pMean, const Ipp32f* pVar, int step, int width, const Ipp32f*
   pVal, const Ipp8u* pSign, int height, Ipp32f* pResult, int frames,
   Ipp32f none);

IppStatus ippsLogGaussMixture_Select_32f_D2L(const Ipp32f** mSrc, const
   Ipp32f** mMean, const Ipp32f** mVar, int width, const Ipp32f* pVal,
   const Ipp8u* pSign, int height, Ipp32f* pResult, int frames, Ipp32f
   none);

IppStatus ippsLogGaussMixture_SelectIdVarScaled_16s32f_D2(const Ipp16s*
   pSrc, const Ipp16s* pMean, int step, int width, const Ipp32s* pVal,
   const Ipp8u* pSign, int height, Ipp32s* pResult, int frames, Ipp32f
   none, int scaleFactor);

IppStatus ippsLogGaussMixture_SelectIdVarScaled_16s32f_D2L(const
   Ipp16s** mSrc, const Ipp16s** mMean, int width, const Ipp32s* pVal,
   const Ipp8u* pSign, int height, Ipp32s* pResult, int frames, Ipp32f
   none, int scaleFactor);

IppStatus ippsLogGaussMixture_SelectIdVar_32f_D2(const Ipp32f* pSrc,
   const Ipp32f* pMean, int step, int width, const Ipp32f* pVal, const
   Ipp8u* pSign, int height, Ipp32f* pResult, int frames, Ipp32f none);

IppStatus ippsLogGaussMixture_SelectIdVar_32f_D2L(const Ipp32f** mSrc,
   const Ipp32f** mMean, int width, const Ipp32f* pVal, const Ipp8u*
   pSign, int height, Ipp32f* pResult, int frames, Ipp32f none);
```

BuildSignTable

Fills sign table for Gaussian mixture calculation.

```
IppStatus ippsBuildSignTable_8u1u(const Ipp32s* pIndx, int num, const
   Ipp8u** mShortlist, int clust, int width, int shift, Ipp8u* pSign,
   int frames, int comps);

IppStatus ippsBuildSignTable_Var_8u1u(const Ipp32s* pIndx, const int*
   pNum, const Ipp8u** mShortlist, int clust, int width, int shift,
   Ipp8u* pSign, int frames, int comps);
```

FillShortlist_Row

Fills row-wise shortlist table for Gaussian selection.

```
IppStatus ippsFillShortlist_Row_1u(const Ipp32s* pIndx, int height, int
   num, Ipp8u** mShortlist, int clust, int width, int shift);
```

```
IppStatus ippsFillShortlist_RowVar_1u(const Ipp32s* pIndx, const int*
    pNum, int height, Ipp8u** mShortlist, int clust, int width, int
    shift);
```

FillShortlist_Column

Fills column-wise shortlist table for Gaussian selection.

```
IppStatus ippsFillShortlist_Column_1u(const Ipp32s* pIndx, int num,
    Ipp8u** mShortlist, int clust, int width, int shift, int height);
IppStatus ippsFillShortlist_ColumnVar_1u(const Ipp32s* pIndx, const int*
    pNum, Ipp8u** mShortlist, int clust, int width, int shift, int
    height);
```

DTW

Computes the distance between observation and reference vector sequences using Dynamic Time Warping algorithm.

```
IppStatus ippsDTW_L2_8u32s_D2Sfs(const Ipp8u* pSrc1, int height1, const
    Ipp8u* pSrc2, int height2, int width, int step, Ipp32s* pDist, int
    delta, Ipp32s beam, int scaleFactor);
IppStatus ippsDTW_L2_8u32s_D2LSfs(const Ipp8u** mSrc1, int height1,
    const Ipp8u** mSrc2, int height2, int width, Ipp32s* pDist, int
    delta, Ipp32s beam, int scaleFactor);
IppStatus ippsDTW_L2Low_16s32s_D2Sfs(const Ipp16s* pSrc1, int height1,
    const Ipp16s* pSrc2, int height2, int width, int step, Ipp32s* pDist,
    int delta, Ipp32s beam, int scaleFactor);
IppStatus ippsDTW_L2Low_16s32s_D2LSfs(const Ipp16s** mSrc1, int height1,
    const Ipp16s** mSrc2, int height2, int width, Ipp32s* pDist, int
    delta, Ipp32s beam, int scaleFactor);
IppStatus ippsDTW_L2_32f_D2(const Ipp32f* pSrc1, int height1, const
    Ipp32f* pSrc2, int height2, int width, int step, Ipp32f* pDist, int
    delta, Ipp32f beam);
IppStatus ippsDTW_L2_32f_D2L(const Ipp32f** mSrc1, int height1, const
    Ipp32f** mSrc2, int height2, int width, Ipp32f* pDist, int delta,
    Ipp32f beam);
```

Model Estimation

MeanColumn

Computes the mean values for the column elements.

```
IppStatus ippsMeanColumn_16s_D2(const Ipp16s* pSrc, int height, int step,
    Ipp16s* pDstMean, int width);
IppStatus ippsMeanColumn_16s_D2L(const Ipp16s** mSrc, int height,
    Ipp16s* pDstMean, int width);
IppStatus ippsMeanColumn_32f_D2(const Ipp32f* pSrc, int height, int step,
    Ipp32f* pDstMean, int width);
```

```
IppStatus ippsMeanColumn_32f_D2L(const Ipp32f** mSrc, int height,
   Ipp32f* pDstMean, int width);
```

VarColumn

Calculates the variances for the column elements.

```
IppStatus ippsVarColumn_16s_D2Sfs(const Ipp16s* pSrc, int height, int
   step, Ipp16s* pSrcMean, Ipp16s* pDstVar, int width, int scaleFactor);
IppStatus ippsVarColumn_16s_D2LSfs(const Ipp16s** mSrc, int height,
   Ipp16s* pSrcMean, Ipp16s* pDstVar, int width, int scaleFactor);
IppStatus ippsVarColumn_32f_D2(const Ipp32f* pSrc, int height, int step,
   Ipp32f* pSrcMean, Ipp32f* pDstVar, int width);
IppStatus ippsVarColumn_32f_D2L(const Ipp32f** mSrc, int height, Ipp32f*
   pSrcMean, Ipp32f* pDstVar, int width);
```

MeanVarColumn

Calculates the means and variances for the column elements of a matrix.

```
IppStatus ippsMeanVarColumn_16s_D2Sfs(const Ipp16s* pSrc, int height,
   int step, Ipp16s* pDstMean, Ipp16s* pDstVar, int width, int
   scaleFactor);
IppStatus ippsMeanVarColumn_16s_D2LSfs(const Ipp16s** mSrc, int height,
   Ipp16s* pDstMean, Ipp16s* pDstVar, int width, int scaleFactor);
IppStatus ippsMeanVarColumn_16s32s_D2Sfs(const Ipp16s* pSrc, int height,
   int step, Ipp32s* pDstMean, Ipp32s* pDstVar, int width, int
   scaleFactor);
IppStatus ippsMeanVarColumn_16s32s_D2LSfs(const Ipp16s** mSrc, int
   height, Ipp32s* pDstMean, Ipp32s* pDstVar, int width, int
   scaleFactor);
IppStatus ippsMeanVarColumn_32f_D2(const Ipp32f* pSrc, int height, int
   step, Ipp32f* pDstMean, Ipp32f* pDstVar, int width);
IppStatus ippsMeanVarColumn_32f_D2L(const Ipp32f** mSrc, int height,
   Ipp32f* pDstMean, Ipp32f* pDstVar, int width);
IppStatus ippsMeanVarColumn_16s16s32s_D2(const Ipp16s* pSrc, int height,
   int step, Ipp16s* pDstMean, Ipp32s* pDstVar, int width);
IppStatus ippsMeanVarColumn_16s16s32s_D2L(const Ipp16s** mSrc, int
   height, Ipp16s* pDstMean, Ipp32s* pDstVar, int width);
```

WeightedMeanColumn

Computes the weighted mean values for the column elements.

```
IppStatus ippsWeightedMeanColumn_32f_D2(const Ipp32f* pSrc, int step,
   const Ipp32f* pWgt, int height, Ipp32f* pDstMean, int width);
IppStatus ippsWeightedMeanColumn_32f_D2L(const Ipp32f** mSrc, const
   Ipp32f* pWgt, int height, Ipp32f* pDstMean, int width);
IppStatus ippsWeightedMeanColumn_64f_D2(const Ipp64f* pSrc, int step,
   const Ipp64f* pWgt, int height, Ipp64f* pDstMean, int width);
```

```
IppStatus ippsWeightedMeanColumn_64f_D2L(const Ipp64f** mSrc, const
   Ipp64f* pWgt, int height, Ipp64f* pDstMean, int width);
```

WeightedVarColumn

Computes the weighted variance values for the column elements.

```
IppStatus ippsWeightedVarColumn_32f_D2(const Ipp32f* pSrc, int step,
   const Ipp32f* pWgt, int height, const Ipp32f* pSrcMean, Ipp32f*
   pDstVar, int width);

IppStatus ippsWeightedVarColumn_32f_D2L(const Ipp32f** mSrc, const
   Ipp32f* pWgt, int height, const Ipp32f* pSrcMean, Ipp32f* pDstVar,
   int width);

IppStatus ippsWeightedVarColumn_64f_D2(const Ipp64f* pSrc, int step,
   const Ipp64f* pWgt, int height, const Ipp64f* pSrcMean, Ipp64f*
   pDstVar, int width);

IppStatus ippsWeightedVarColumn_64f_D2L(const Ipp64f** mSrc, const
   Ipp64f* pWgt, int height, const Ipp64f* pSrcMean, Ipp64f* pDstVar,
   int width);
```

WeightedMeanVarColumn

Computes weighted mean and variance values for the column elements.

```
IppStatus ippsWeightedMeanVarColumn_32f_D2(const Ipp32f* pSrc, int step,
   const Ipp32f* pWgt, int height, Ipp32f* pDstMean, Ipp32f* pDstVar,
   int width);

IppStatus ippsWeightedMeanVarColumn_32f_D2L(const Ipp32f** mSrc, const
   Ipp32f* pWgt, int height, Ipp32f* pDstMean, Ipp32f* pDstVar, int
   width);

IppStatus ippsWeightedMeanVarColumn_64f_D2(const Ipp64f* pSrc, int step,
   const Ipp64f* pWgt, int height, Ipp64f* pDstMean, Ipp64f* pDstVar,
   int width);

IppStatus ippsWeightedMeanVarColumn_64f_D2L(const Ipp64f** mSrc, const
   Ipp64f* pWgt, int height, Ipp64f* pDstMean, Ipp64f* pDstVar, int
   width);
```

NormalizeColumn

Normalizes the matrix columns given the column means and variances.

```
IppStatus ippsNormalizeColumn_16s_D2Sfs(Ipp16s* pSrcDst, int step, int
   height, const Ipp16s* pMean, const Ipp16s* pVar, int width, int
   scaleFactor);

IppStatus ippsNormalizeColumn_16s_D2LSfs(Ipp16s** mSrcDst, int height,
   const Ipp16s* pMean, const Ipp16s* pVar, int width, int scaleFactor);

IppStatus ippsNormalizeColumn_32f_D2(Ipp32f* pSrcDst, int step, int
   height, const Ipp32f* pMean, const Ipp32f* pVar, int width);

IppStatus ippsNormalizeColumn_32f_D2L(Ipp32f** mSrcDst, int height,
   const Ipp32f* pMean, const Ipp32f* pVar, int width);
```

NormalizeInRange

Normalizes and scales input vector elements.

```
IppStatus ippsNormalizeInRange_16s8u(const Ipp16s* pSrc, Ipp8u* pDst,
   int len, Ipp32f lowCut, Ipp32f highCut, Ipp8u range);
IppStatus ippsNormalizeInRange_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, Ipp32f lowCut, Ipp32f highCut, Ipp16s range);
IppStatus ippsNormalizeInRange_16s_I(Ipp16s* pSrcDst, int len, Ipp32f
   lowCut, Ipp32f highCut, Ipp16s range);
IppStatus ippsNormalizeInRange_32f8u(const Ipp32f* pSrc, Ipp8u* pDst,
   int len, Ipp32f lowCut, Ipp32f highCut, Ipp8u range);
IppStatus ippsNormalizeInRange_32f16s(const Ipp32f* pSrc, Ipp16s* pDst,
   int len, Ipp32f lowCut, Ipp32f highCut, Ipp16s range);
IppStatus ippsNormalizeInRange_32f(const Ipp32f* pSrc, Ipp32f* pDst, int
   len, Ipp32f lowCut, Ipp32f highCut, Ipp32f range);
IppStatus ippsNormalizeInRange_32f_I(Ipp32f* pSrcDst, int len, Ipp32f
   lowCut, Ipp32f highCut, Ipp32f range);
IppStatus ippsNormalizeInRangeMinMax_16s8u(const Ipp16s* pSrc, Ipp8u*
   pDst, int len, Ipp16s valMin, Ipp16s valMax, Ipp32f lowCut, Ipp32f
   highCut, Ipp8u range);
IppStatus ippsNormalizeInRangeMinMax_16s(const Ipp16s* pSrc, Ipp16s*
   pDst, int len, Ipp16s valMin, Ipp16s valMax, Ipp32f lowCut, Ipp32f
   highCut, Ipp16s range);
IppStatus ippsNormalizeInRangeMinMax_16s_I(Ipp16s* pSrcDst, int len,
   Ipp16s valMin, Ipp16s valMax, Ipp32f lowCut, Ipp32f highCut, Ipp16s
   range);
IppStatus ippsNormalizeInRangeMinMax_32f8u(const Ipp32f* pSrc, Ipp8u*
   pDst, int len, Ipp32f valMin, Ipp32f valMax, Ipp32f lowCut, Ipp32f
   highCut, Ipp8u range);
IppStatus ippsNormalizeInRangeMinMax_32f16s(const Ipp32f* pSrc, Ipp16s*
   pDst, int len, Ipp32f valMin, Ipp32f valMax, Ipp32f lowCut, Ipp32f
   highCut, Ipp16s range);
IppStatus ippsNormalizeInRangeMinMax_32f(const Ipp32f* pSrc, Ipp32f*
   pDst, int len, Ipp32f valMin, Ipp32f valMax, Ipp32f lowCut, Ipp32f
   highCut, Ipp32f range);
IppStatus ippsNormalizeInRangeMinMax_32f_I(Ipp32f* pSrcDst, int len,
   Ipp32f valMin, Ipp32f valMax, Ipp32f lowCut, Ipp32f highCut, Ipp32f
   range);
```

MeanVarAcc

Accumulates the estimates for the mean and variance re-estimation.

```
IppStatus ippsMeanVarAcc_32f(Ipp32f const* pSrc, Ipp32f const* pSrcMean,
   Ipp32f* pDstMeanAcc, Ipp32f* pDstVarAcc, int len, Ipp32f val);
IppStatus ippsMeanVarAcc_64f(Ipp64f const* pSrc, Ipp64f const* pSrcMean,
   Ipp64f* pDstMeanAcc, Ipp64f* pDstVarAcc, int len, Ipp64f val);
```

GaussianDist

Calculates the distance between two Gaussians.

```
IppStatus ippsGaussianDist_32f(const Ipp32f* pMean1, const Ipp32f*
   pVar1, const Ipp32f* pMean2, const Ipp32f* pVar2, int len, Ipp32f*
   pResult, Ipp32f wgt1, Ipp32f det1, Ipp32f wgt2, Ipp32f det2);

IppStatus ippsGaussianDist_64f(const Ipp64f* pMean1, const Ipp64f*
   pVar1, const Ipp64f* pMean2, const Ipp64f* pVar2, int len, Ipp64f*
   pResult, Ipp64f wgt1, Ipp64f det1, Ipp64f wgt2, Ipp64f det2);
```

GaussianSplit

Splits a single Gaussian component into two with the same variance.

```
IppStatus ippsGaussianSplit_32f(Ipp32f* pMean1, Ipp32f* pVar1, Ipp32f*
   pMean2, Ipp32f* pVar2, int len, Ipp32f val);

IppStatus ippsGaussianSplit_64f(Ipp64f* pMean1, Ipp64f* pVar1, Ipp64f*
   pMean2, Ipp64f* pVar2, int len, Ipp64f val);
```

GaussianMerge

Merges two Gaussian probability distribution functions.

```
IppStatus ippsGaussianMerge_32f(const Ipp32f* pMean1, const Ipp32f*
   pVar1, const Ipp32f* pMean2, const Ipp32f* pVar2, Ipp32f* pDstMean,
   Ipp32f* pDstVar, int len, Ipp32f* pDstDet, Ipp32f wgt1, Ipp32f wgt2);

IppStatus ippsGaussianMerge_64f(const Ipp64f* pMean1, const Ipp64f*
   pVar1, const Ipp64f* pMean2, const Ipp64f* pVar2, Ipp64f* pDstMean,
   Ipp64f* pDstVar, int len, Ipp64f* pDstDet, Ipp64f wgt1, Ipp64f wgt2);
```

Entropy

Calculates entropy of the input vector.

```
IppStatus ippsEntropy_32f(const Ipp32f* pSrc, int len, Ipp32f* pResult);

IppStatus ippsEntropy_16s32s_Sfs(const Ipp16s* pSrc, int srcShiftVal,
   int len, Ipp32s* pResult, int scaleFactor);
```

SinC

Calculates sine divided by its argument.

```
IppStatus ippsSinc_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);

IppStatus ippsSinc_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);

IppStatus ippsSinc_32f64f(const Ipp32f* pSrc, Ipp64f* pDst, int len);

IppStatus ippsSinc_32f_I(Ipp32f* pSrcDst, int len);

IppStatus ippsSinc_64f_I(Ipp64f* pSrcDst, int len);
```

ExpNegSqr

Calculates exponential of the squared argument taken with the inverted sign.

```
IppStatus ippsExpNegSqr_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);

IppStatus ippsExpNegSqr_64f(const Ipp64f* pSrc, Ipp64f* pDst, int len);

IppStatus ippsExpNegSqr_32f64f(const Ipp32f* pSrc, Ipp64f* pDst, int
   len);
```

```
IppStatus ippsExpNegSqr_32f_I(Ipp32f* pSrcDst, int len);
IppStatus ippsExpNegSqr_64f_I(Ipp64f* pSrcDst, int len);
```

BhatDist

Calculates the Bhattacharia distance between two Gaussians.

```
IppStatus ippsBhatDist_32f(const Ipp32f* pMean1, const Ipp32f* pVar1,
   const Ipp32f* pMean2, const Ipp32f* pVar2, int len, Ipp32f* pResult);
IppStatus ippsBhatDist_32f64f(const Ipp32f* pMean1, const Ipp32f* pVar1,
   const Ipp32f* pMean2, const Ipp32f* pVar2, int len, Ipp64f* pResult);
IppStatus ippsBhatDistSLog_32f(const Ipp32f* pMean1, const Ipp32f*
   pVar1, const Ipp32f* pMean2, const Ipp32f* pVar2, int len, Ipp32f*
   pResult, Ipp32f sumLog1, Ipp32f sumLog2);
IppStatus ippsBhatDistSLog_32f64f(const Ipp32f* pMean1, const Ipp32f*
   pVar1, const Ipp32f* pMean2, const Ipp32f* pVar2, int len, Ipp64f*
   pResult, Ipp32f sumLog1, Ipp32f sumLog2);
```

UpdateMean

Updates the mean vector in the EM training algorithm.

```
IppStatus ippsUpdateMean_32f(const Ipp32f* pMeanAcc, Ipp32f* pMean, int
   len, Ipp32f meanOcc);
IppStatus ippsUpdateMean_64f(const Ipp64f* pMeanAcc, Ipp64f* pMean, int
   len, Ipp64f meanOcc);
```

UpdateVar

Updates the variance vector in the EM training algorithm.

```
IppStatus ippsUpdateVar_32f(const Ipp32f* pMeanAcc, const Ipp32f*
   pVarAcc, const Ipp32f* pVarFloor, Ipp32f* pVar, int len, Ipp32f
   meanOcc, Ipp32f varOcc);
IppStatus ippsUpdateVar_64f(const Ipp64f* pMeanAcc, const Ipp64f*
   pVarAcc, const Ipp64f* pVarFloor, Ipp64f* pVar, int len, Ipp64f
   meanOcc, Ipp64f varOcc);
```

UpdateWeight

Updates the weight values of Gaussian mixtures in the EM training algorithm.

```
IppStatus ippsUpdateWeight_32f(const Ipp32f* pWgtAcc, Ipp32f* pWgt, int
   len, Ipp32f* pWgtSum, Ipp32f wgtOcc, Ipp32f wgtThresh);
IppStatus ippsUpdateWeight_64f(const Ipp64f* pWgtAcc, Ipp64f* pWgt, int
   len, Ipp64f* pWgtSum, Ipp64f wgtOcc, Ipp64f wgtThresh);
```

UpdateGConst

Updates the fixed constant in the Gaussian output probability density function.

```
IppStatus ippsUpdateGConst_32f(const Ipp32f* pVar, int len, Ipp32f*
   pDet);
IppStatus ippsUpdateGConst_64f(const Ipp64f* pVar, int len, Ipp64f*
   pDet);
```

```
IppStatus ippsUpdateGConst_DirectVar_32f(const Ipp32f* pVar, int len,
   Ipp32f* pDet);
IppStatus ippsUpdateGConst_DirectVar_64f(const Ipp64f* pVar, int len,
   Ipp64f* pDet);
```

OutProbPreCalc

Pre-calculates the part of Gaussian mixture output probability that is irrelevant to observation vectors.

```
IppStatus ippsOutProbPreCalc_32s(const Ipp32s* pWeight, const Ipp32s*
   pSrc, Ipp32s* pDst, int len);
IppStatus ippsOutProbPreCalc_32s_I(const Ipp32s* pWeight, Ipp32s*
   pSrcDst, int len);
IppStatus ippsOutProbPreCalc_32f(const Ipp32f* pWeight, const Ipp32f*
   pSrc, Ipp32f* pDst, int len);
IppStatus ippsOutProbPreCalc_64f(const Ipp64f* pWeight, const Ipp64f*
   pSrc, Ipp64f* pDst, int len);
IppStatus ippsOutProbPreCalc_32f_I(const Ipp32f* pWeight, Ipp32f*
   pSrcDst, int len);
IppStatus ippsOutProbPreCalc_64f_I(const Ipp64f* pWeight, Ipp64f*
   pSrcDst, int len);
```

DcsClustLAccumulate

Updates the accumulators for calculating the state-cluster likelihood in the decision-tree clustering algorithm.

```
IppStatus ippsDcsClustLAccumulate_32f(const Ipp32f* pMean, const Ipp32f*
   pVar, Ipp32f* pDstSum, Ipp32f* pDstSqr, int len, Ipp32f occ);
IppStatus ippsDcsClustLAccumulate_64f(const Ipp64f* pMean, const Ipp64f*
   pVar, Ipp64f* pDstSum, Ipp64f* pDstSqr, int len, Ipp64f occ);
IppStatus ippsDcsClustLAccumulate_DirectVar_32f(const Ipp32f* pMean,
   const Ipp32f* pVar, Ipp32f* pDstSum, Ipp32f* pDstSqr, int len, Ipp32f
   occ);
IppStatus ippsDcsClustLAccumulate_DirectVar_64f(const Ipp64f* pMean,
   const Ipp64f* pVar, Ipp64f* pDstSum, Ipp64f* pDstSqr, int len, Ipp64f
   occ);
```

DcsClustLCompute

Calculates the likelihood of an HMM state cluster in the decision-tree state-clustering algorithm.

```
IppStatus ippsDcsClustLCompute_64f(const Ipp64f* pSrcSum, const Ipp64f*
   pSrcSqr, int len, Ipp64f* pDst, Ipp64f occ);
IppStatus ippsDcsClustLCompute_32f64f(const Ipp32f* pSrcSum, const
   Ipp32f* pSrcSqr, int len, Ipp64f* pDst, Ipp32f occ);
```

Model Adaptation

AddMulColumn
Adds a weighted matrix column to the other column.

```
IppStatus ippsAddMulColumn_64f_D2L(Ipp64f** mSrcDst, int width, int
   height, int col1, int col2, int row1, const Ipp64f val);
```

AddMulRow
Adds a weighted vector to the other vector.

```
ippsAddMulRow_64f(const Ipp64f* pSrc, Ipp64f* pSrcDst, int len,
   const Ipp64f val);
```

QRTransColumn
Performs the QR transformation.

```
IppStatus ippsQRTransColumn_64f_D2L(Ipp64f** mSrcDst, int width, int
   height, int col1, int col2, const Ipp64f val1, const Ipp64f val2);
```

DotProdColumn
Calculates the dot product of two matrix columns.

```
IppStatus ippsDotProdColumn_64f_D2L(const Ipp64f** mSrc, int width,
   int height, Ipp64f* pSum, int col1, int col2, int row1);
```

MulColumn
Multiplies a matrix column by a value.

```
IppStatus ippsMulColumn_64f_D2L(Ipp64f** mSrcDst, int width, int height,
   int col1, int row1, const Ipp64f val);
```

SumColumnAbs
Calculates the absolute sum of matrix column elements.

```
IppStatus ippsSumColumnAbs_64f_D2L(const Ipp64f** mSrc, int width,
   int height, Ipp64f* pSum, int col1, int row1);
```

SumColumnSqr
Calculates the square sums of weighted matrix column elements.

```
IppStatus ippsSumColumnSqr_64f_D2L(Ipp64f** mSrcDst, int width, int
   height,Ipp64f* pSum, int col1, int row1, const Ipp64f val);
```

SumRowAbs
Calculates the absolute sum of the vector elements.

```
IppStatus ippsSumRowAbs_64f(const Ipp64f* pSrc, int len, Ipp64f* pSum);
```

SumRowSqr
Calculates the square sum of weighted vector elements.

```
IppStatus ippsSumRowSqr_64f(Ipp64f* pSrcDst, int len, Ipp64f* pSum,
   const Ipp64f val);
```

SVD, SVDSort

Performs single value decomposition on a matrix.

```
IppStatus ippsSVD_64f_D2(const Ipp64f* pSrcA, Ipp64f* pDstU, int height,
   Ipp64f* pDstW, Ipp64f* pDstV, int width, int step, int nIter);
IppStatus ippsSVD_64f_D2L(const Ipp64f** mSrcA, Ipp64f** mDstU, int
   height, Ipp64f* pDstW, Ipp64f** mDstV, int width, int nIter);
IppStatus ippsSVD_64f_D2_I(Ipp64f* pSrcDstA, int height, Ipp64f* pDstW,
   Ipp64f* pDstV, int width, int step, int nIter);
IppStatus ippsSVD_64f_D2L_I(Ipp64f** mSrcDstA, int height, Ipp64f*
   pDstW, Ipp64f** mDstV, int width, int nIter);
IppStatus ippsSVDSort_64f_D2(const Ipp64f* pSrcA, Ipp64f* pDstU, int
   height, Ipp64f* pDstW, Ipp64f* pDstV, int width, int step, int
   nIter);
IppStatus ippsSVDSort_64f_D2L(const Ipp64f** mSrcA, Ipp64f** mDstU, int
   height, Ipp64f* pDstW, Ipp64f** mDstV, int width, int nIter);
IppStatus ippsSVDSort_64f_D2_I(Ipp64f* pSrcDstA, int height, Ipp64f*
   pDstW, Ipp64f* pDstV, int width, int step, int nIter);
IppStatus ippsSVDSort_64f_D2L_I(Ipp64f** mSrcDstA, int height, Ipp64f*
   pDstW, Ipp64f** mDstV, int width, int nIter);
```

WeightedSum

Calculates the weighted sums of two input vector elements.

```
IppStatus ippsWeightedSum_16s(const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, Ipp32f weight1, Ipp32f weight2);
IppStatus ippsWeightedSum_32f(const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len, Ipp32f weight1, Ipp32f weight2);
IppStatus ippsWeightedSum_64f(const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len, Ipp64f weight1, Ipp64f weight2);
IppStatus ippsWeightedSumHalf_16s(const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pDst, int len, Ipp32f weight1, Ipp32f weight2);
IppStatus ippsWeightedSumHalf_32f(const Ipp32f* pSrc1, const Ipp32f*
   pSrc2, Ipp32f* pDst, int len, Ipp32f weight1, Ipp32f weight2);
IppStatus ippsWeightedSumHalf_64f(const Ipp64f* pSrc1, const Ipp64f*
   pSrc2, Ipp64f* pDst, int len, Ipp64f weight1, Ipp64f weight2);
```

Vector Quantization

FormVector

Constructs an output vector of multiple streams from codebook entries.

```
IppStatus ippsFormVector_8u16s(const Ipp8u* pInd, const Ipp16s** mSrc,
   const Ipp32s* pHeights, const Ipp32s* pWidths, const Ipp32s* pSteps,
   int nStream, Ipp16s* pDst);

IppStatus ippsFormVector_16s16s(const Ipp16s* pInd, const Ipp16s** mSrc,
   const Ipp32s* pHeights, const Ipp32s* pWidths, const Ipp32s* pSteps,
   int nStream, Ipp16s* pDst);

IppStatus ippsFormVector_8u32f(const Ipp8u* pInd, const Ipp32f** mSrc,
   const Ipp32s* pHeights, const Ipp32s* pWidths, const Ipp32s* pSteps,
   int nStream, Ipp32f*  pDst);

IppStatus ippsFormVector_16s32f(const Ipp16s* pInd, const Ipp32f** mSrc,
   const Ipp32s* pHeights, const Ipp32s* pWidths, const Ipp32s* pSteps,
   int nStream, Ipp32f*  pDst);


IppStatus ippsFormVector_2i_8u16s(const Ipp8u* pInd, const Ipp16s**
   mSrc, const Ipp32s* pHeights, Ipp16s* pDst, int len);

IppStatus ippsFormVector_2i_16s16s(const Ipp16s* pInd, const Ipp16s**
   mSrc, const Ipp32s* pHeights, Ipp16s* pDst, int len);

IppStatus ippsFormVector_2i_8u32f(const Ipp8u* pInd, const Ipp32f**
   mSrc, const Ipp32s* pHeights, Ipp32f* pDst, int len);

IppStatus ippsFormVector_2i_16s32f(const Ipp16s* pInd, const Ipp32f**
   mSrc, const Ipp32s* pHeights, Ipp32f* pDst, int len);


IppStatus ippsFormVector_4i_8u16s(const Ipp8u* pInd, const Ipp16s**
   mSrc, const Ipp32s* pHeights, Ipp16s* pDst, int len);

IppStatus ippsFormVector_4i_16s16s(const Ipp16s* pInd, const Ipp16s**
   mSrc, const Ipp32s* pHeights, Ipp16s* pDst, int len);

IppStatus ippsFormVector_4i_8u32f(const Ipp8u* pInd, const Ipp32f**
   mSrc, const Ipp32s* pHeights, Ipp32f* pDst, int len);

IppStatus ippsFormVector_4i_16s32f(const Ipp16s* pInd, const Ipp32f**
   mSrc, const Ipp32s* pHeights, Ipp32f* pDst, int len);
```

CdbkGetSize

Calculates the size in bytes of the codebook.

```
IppStatus ippsCdbkGetSize_16s(int width, int step, int height, int
   cdbkSize, IppCdbk_Hint hint, int* pSize);
```

CdbkInit

Initializes the structure that contains the codebook.

```
IppStatus ippsCdbkInit_L2_16s(IppsCdbkState_16s* pCdbk, const Ipp16s*
   pSrc, int width, int step, int height, int cdbkSize, Ipp_Cdbk_Hint
   hint);
```

CdbkInitAlloc

Initializes the codebook structure.

```
IppStatus ippsCdbkInitAlloc_L2_16s(IppsCdbkState_16s** pCdbk, const
   Ipp16s* pSrc, int width, int step, int height, int cdbkSize,
   Ipp_Cdbk_Hint hint);
IppStatus ippsCdbkInitAlloc_L2_32f(IppsCdbkState_32f** pCdbk, const
   Ipp32f* pSrc, int width, int step, int height, int cdbkSize,
   Ipp_Cdbk_Hint hint);
IppStatus ippsCdbkInitAlloc_WgtL2_16s(IppsCdbkState_16s** pCdbk, const
   Ipp16s* pSrc, const Ipp16s* pWgt, int width, int step, int height, int
   cdbkSize, Ipp_Cdbk_Hint hint);
IppStatus ippsCdbkInitAlloc_WgtL2_32f(IppsCdbkState_32f** pCdbk, const
   Ipp32f* pSrc, const Ipp32f* pWgt, int width, int step, int height, int
   cdbkSize, Ipp_Cdbk_Hint hint);
```

CdbkFree

Destroys the codebook structure.

```
IppStatus ippsCdbkFree_16s(IppsCdbkState_16s* pCdbk);
IppStatus ippsCdbkFree_32f(IppsCdbkState_32f* pCdbk);
```

GetCdbkSize

Retrieves the number of codevectors in the codebook.

```
IppStatus ippsGetCdbkSize_16s(const IppsCdbkState_16s* pCdbk, int*
   pNum);
IppStatus ippsGetCdbkSize_32f(const IppsCdbkState_32f* pCdbk, int*
   pNum);
```

GetCodebook

Retrieves the codevectors from the codebook.

```
IppStatus ippsGetCodebook_16s(const IppsCdbkState_16s* pCdbk, Ipp16s*
   pDst, int step);
IppStatus ippsGetCodebook_32f(const IppsCdbkState_32f* pCdbk, Ipp32f*
   pDst, int step);
```

VQ

Quantizes the input vectors given a codebook.

```
IppStatus ippsVQ_16s(const Ipp16s* pSrc, int step, Ipp32s* pIndx, int
   height, const IppsCdbkState_16s* pCdbk);
IppStatus ippsVQ_32f(const Ipp32f* pSrc, int step, Ipp32s* pIndx, int
   height, const IppsCdbkState_32f* pCdbk);
```

```
IppStatus ippsVQDist_16s32s_Sfs(const Ipp16s* pSrc, int step, Ipp32s* pIndx,
   Ipp32s* pDist, int height, const IppsCdbkState_16s* pCdbk,
   int scaleFactor);

IppStatus ippsVQDist_32f(const Ipp32f* pSrc, int step, Ipp32s* pIndx,
   Ipp32f* pDist, int height, const IppsCdbkState_32f* pCdbk);
```

VQSingle_Sort, VQSingle_Thresh

Quantizes the input vector given a codebook and gets several closest clusters.

```
IppStatus ippsVQSingle_Sort_32f(const Ipp32f* pSrc, Ipp32s* pIndx, const
   IppsCdbkState_32f* pCdbk, int num);

IppStatus ippsVQSingle_Sort_16s(const Ipp16s* pSrc, Ipp32s* pIndx, const
   IppsCdbkState_16s* pCdbk, int num);

IppStatus ippsVQDistSingle_Sort_32f(const Ipp32f* pSrc, Ipp32s* pIndx,
   Ipp32f* pDist, const IppsCdbkState_32f* pCdbk, int num);

IppStatus ippsVQDistSingle_Sort_16s32s_Sfs(const Ipp16s* pSrc, Ipp32s* pIndx,
   Ipp32s* pDist, const IppsCdbkState_16s* pCdbk, int num, int scaleFactor);

IppStatus ippsVQSingle_Thresh_32f(const Ipp32f* pSrc, Ipp32s* pIndx, const
   IppsCdbkState_32f* pCdbk, Ipp32f val, int* pnum);

IppStatus ippsVQSingle_Thresh_16s(const Ipp16s* pSrc, Ipp32s* pIndx, const
   IppsCdbkState_16s* pCdbk, Ipp32f val, int* pnum);

IppStatus ippsVQDistSingle_Thresh_32f(const Ipp32f* pSrc, Ipp32s* pIndx,
   Ipp32f* pDist, const IppsCdbkState_32f* pCdbk, Ipp32f val, int* pnum);

IppStatus ippsVQDistSingle_Thresh_16s32s_Sfs(const Ipp16s* pSrc, Ipp32s*
   pIndx, Ipp32s* pDist, const IppsCdbkState_16s* pCdbk, Ipp32f val, int*
   pnum, int scaleFactor);
```

SplitVQ

Quantizes a multiple-stream vector given the codebooks.

```
IppStatus ippsSplitVQ_16s16s(const Ipp16s* pSrc, int srcStep, Ipp16s* pDst,
   int dstStep, int height, const IppsCdbkState_16s** pCdbks, int nStream);

IppStatus ippsSplitVQ_16s8u(const Ipp16s* pSrc, int srcStep, Ipp8u* pDst, int
   dstStep, int height, const IppsCdbkState_16s** pCdbks, int nStream);

IppStatus ippsSplitVQ_16s1u(const Ipp16s* pSrc, int srcStep, Ipp8u* pDst, int
   dstBitStep, int height, const IppsCdbkState_16s** pCdbks, int nStream);

IppStatus ippsSplitVQ_32f16s(const Ipp32f* pSrc, int srcStep, Ipp16s* pDst,
   int dstStep, int height, const IppsCdbkState_32f** pCdbks, int nStream);

IppStatus ippsSplitVQ_32f8u(const Ipp32f* pSrc, int srcStep, Ipp8u* pDst, int
   dstStep, int height, const IppsCdbkState_32f** pCdbks, int nStream);

IppStatus ippsSplitVQ _32f1u(const Ipp32f* pSrc, int srcStep, Ipp8u* pDst,
   int dstBitStep, int height, const IppsCdbkState_32f** pCdbks, int
   nStream);
```

FormVectorVQ

Constructs multiple-stream vectors from codebooks, given indexes.

```
IppStatus ippsFormVectorVQ_16s16s(const Ipp16s* pSrc, int srcStep, Ipp16s*
    pDst, int dstStep, int height, const IppsCdbkState_16s** pCdbks, int
    nStream);

IppStatus ippsFormVectorVQ_8u16s(const Ipp8u* pSrc, int srcStep, Ipp16s*
    pDst, int dstStep, int height, const IppsCdbkState_16s** pCdbks, int
    nStream);

IppStatus ippsFormVectorVQ_1u16s(const Ipp8u* pSrc, int srcBitStep,
    Ipp16s* pDst, int dstStep, int height, const IppsCdbkState_16s**
    pCdbks, int nStream);

IppStatus ippsFormVectorVQ_16s32f(const Ipp16s* pSrc, int srcStep, Ipp32f*
    pDst, int dstStep, int height, const IppsCdbkState_32f** pCdbks, int
    nStream);

IppStatus ippsFormVectorVQ_8u32f(const Ipp8u* pSrc, int srcStep, Ipp32f*
    pDst, int dstStep, int height, const IppsCdbkState_32f** pCdbks, int
    nStream);

IppStatus ippsFormVectorVQ_1u32f(const Ipp8u* pSrc, int srcBitStep,
    Ipp32f* pDst, int dstStep, int height, const IppsCdbkState_32f**
    pCdbks, int nStream);
```

Polyphase Resampling

ResamplePolyphaseInit

Initializes the structure for polyphase resampling without calculating the filter coefficients.

```
IppStatus ippsResamplePolyphaseFixedInit_16s(
    IppsResamplingPolyphaseFixed_16s* pSpec, int inRate, int outRate, int
    len, IppHintAlgorithm hint);

IppStatus ippsResamplePolyphaseFixedInit_32f(
    IppsResamplingPolyphaseFixed_32f* pSpec, int inRate, int outRate, int
    len, IppHintAlgorithm hint);
```

ResamplePolyphaseGetSize

Gets the size of the polyphase resampling structure.

```
IppStatus ippsResamplePolyphaseFixedGetSize_16s(int inRate, int outRate,
    int len, int* pSize, int* pLen, int* pHeight, IppHintAlgorithm hint);

IppStatus ippsResamplePolyphaseFixedGetSize_32f(int inRate, int outRate,
    int len, int* pSize, int* pLen, int* pHeight, IppHintAlgorithm hint);
```

ResamplePolyphaseSetFilter

Sets polyphase resampling filter coefficients.

```
IppStatus ippsResamplePolyphaseSetFixedFilter_16s(
    IppsResamplingPolyphaseFixed_16s* pSpec, const Ipp16s* pSrc, int step,
    int height);
```

```
IppStatus ippsResamplePolyphaseSetFixedFilter_32f(
   IppsResamplingPolyphaseFixed_32f* pSpec, const Ipp32f* pSrc, int
   step, int height);
```

ResamplePolyphaseGetFilter

Gets polyphase resampling filter coefficients.

```
IppStatus ippsResamplePolyphaseGetFixedFilter_16s(
   IppsResamplingPolyphaseFixed_16s* pSpec, const Ipp16s* pSrc, int
   step, int height);
IppStatus ippsResamplePolyphaseGetFixedFilter_32f(
   IppsResamplingPolyphaseFixed_32f* pSpec, const Ipp32f* pSrc, int
   step, int height);
```

ResamplePolyphaseInitAlloc

Initializes the structure for polyphase data resampling.

```
IppStatus
   ippsResamplePolyphaseInitAlloc_16s(IppsResamplingPolyphase_16s**
   pSpec, Ipp32f window, int nStep, Ipp32f rollf, Ipp32f alpha,
   IppHintAlgorithm hint);
IppStatus
   ippsResamplePolyphaseInitAlloc_32f(IppsResamplingPolyphase_32f**
   pSpec, Ipp32f window, int nStep, Ipp32f rollf, Ipp32f alpha,
   IppHintAlgorithm hint);
IppStatus ippsResamplePolyphaseFixedInitAlloc_16s(
   IppsResamplingPolyphaseFixed_16s** pSpec, int inRate, int outRate,
   int len, Ipp32f rollf, Ipp32f alpha, IppHintAlgorithm hint);
IppStatus ippsResamplePolyphaseFixedInitAlloc_32f(
   IppsResamplingPolyphaseFixed_32f** pSpec, int inRate, int outRate,
   int len, Ipp32f rollf, Ipp32f alpha, IppHintAlgorithm hint);
```

ResamplePolyphaseFree

Frees structure for polyphase data resampling.

```
IppStatus ippsResamplePolyphaseFree_16s(IppsResamplingPolyphase_16s*
   pSpec);
IppStatus ippsResamplePolyphaseFree_32f(IppsResamplingPolyphase_32f*
   pSpec);
IppStatus
   ippsResamplePolyphaseFixedFree_16s(IppsResamplingPolyphaseFixed_16s*
   pSpec);
IppStatus
   ippsResamplePolyphaseFixedFree_32f(IppsResamplingPolyphaseFixed_32f*
   pSpec);
```

ResamplePolyphase

Resamples input data using polyphase filters.

```
IppStatus ippsResamplePolyphase_16s(const IppsResamplingPolyphase_16s*
   pSpec, const Ipp16s* pSrc, int len, Ipp16s* pDst, Ipp64f factor,
   Ipp32f norm, Ipp64f* pTime, int* pOutlen);

IppStatus ippsResamplePolyphase_32f(const IppsResamplingPolyphase_32f*
   pSpec, const Ipp32f* pSrc, int len, Ipp32f* pDst, Ipp64f factor,
   Ipp32f norm, Ipp64f* pTime, int* pOutlen);

IppStatus ippsResamplePolyphaseFixed_16s(const
   IppsResamplingPolyphaseFixed_16s* pSpec, const Ipp16s* pSrc, int len,
   Ipp16s* pDst, Ipp32f norm, Ipp64f* pTime, int* pOutlen);

IppStatus ippsResamplePolyphaseFixed_32f(const
   IppsResamplingPolyphaseFixed_32f* pSpec, const Ipp32f* pSrc, int len,
   Ipp32f* pDst, Ipp32f norm, Ipp64f* pTime, int* pOutlen);
```

Advanced Aurora Functions

SmoothedPowerSpectrumAurora

Calculates smoothed magnitude of the FFT output.

```
IppStatus ippsSmoothedPowerSpectrumAurora_16s(const Ipp16s* pSrc,
   Ipp16s* pDst, int len);

IppStatus ippsSmoothedPowerSpectrumAurora_32f(const Ipp32f* pSrc,
   Ipp32f* pDst, int len);

IppStatus ippsSmoothedPowerSpectrumAurora_32s64s_Sfs(Ipp32s* pSrc,
   Ipp64s* pDst, Ipp32s len, int scaleFactor)

IppStatus  ippsSmoothedPowerSpectrumAurora_32s_Sfs(Ipp32s* pSrc, Ipp32s*
   pDst, Ipp32s len, int scaleFactor)
```

NoiseSpectrumUpdate_Aurora

Updates the noise spectrum.

```
IppStatus ippsNoiseSpectrumUpdate_Aurora_32f(const Ipp32f* pSrc, const
   Ipp32f* pSrcNoise,  Ipp32f* pDst, int len);

IppStatus  ippsNoiseSpectrumUpdate_Aurora_16s_Sfs(const Ipp16s* pSrc,
   const Ipp16s* pSrcNoise, Ipp16s* pDst, int len, int scaleFactor);

IppStatus  ippsNoiseSpectrumUpdate_Aurora_32s_Sfs(const Ipp32s* pSrc,
   const Ipp32s* pSrcNoise, Ipp32s* pDst, int len, int scaleFactor);
```

WienerFilterDesign_Aurora

Calculates an improved transfer function of the adaptive Wiener filter.

```
IppStatus ippsWienerFilterDesign_Aurora_32f(const Ipp32f* pSrc, const
   Ipp32f* pNoise, const Ipp32f* pDen, Ipp32f* pDst, int len);

IppStatus ippsWienerFilterDesign_Aurora_16s(const Ipp16s* pSrc, const
   Ipp16s* pNoise, const Ipp16s* pDen, Ipp16s* pDst, int len);
```

MelFBankInitAlloc_Aurora

Initializes the structure for performing the Mel-frequency filter bank analysis.

```
IppStatus ippsMelFBankInitAllocLow_Aurora_16s(IppsFBankState_16s** pFBank);
IppStatus ippsMelFBankInitAllocLow_Aurora_32f(IppsFBankState_32f** pFBank);
IppStatus ippsMelFBankInitAllocHigh_Aurora_16s(IppsFBankState_16s** pFBank);
IppStatus ippsMelFBankInitAllocHigh_Aurora_32f(IppsFBankState_32f** pFBank);
```

TabsCalculation_Aurora

Calculates filter coefficients for residual filter.

```
IppStatus ippsTabsCalculation_Aurora_16s(const Ipp16s* pSrc, Ipp16s* pDst);
IppStatus ippsTabsCalculation_Aurora_32f(const Ipp32f* pSrc, Ipp32f* pDst);
```

ResidualFilter_Aurora

Calculates a denoised waveform signal.

```
IppStatus ippsResidualFilter_Aurora_16s_Sfs(const Ipp16s* pSrc, Ipp16s*
   pDst, const Ipp16s* pTabs, int scaleFactor);
IppStatus ippsResidualFilter_Aurora_32f(const Ipp32f* pSrc, Ipp32f* pDst,
   const Ipp32f* pTabs);
```

WaveProcessing_Aurora

Processes waveform data after noise reduction.

```
IppStatus ippsWaveProcessing_Aurora_32f(const Ipp32f* pSrc, Ipp32f* pDst);
IppStatus ippsWaveProcessing_Aurora_16s(const Ipp16s* pSrc, Ipp16s* pDst);
```

LowHighFilter_Aurora

Calculates low band and high band filters.

```
IppStatus ippsLowHighFilter_Aurora_16s_Sfs(const Ipp16s* pSrc, Ipp16s*
   pDstLow, Ipp16s* pDstHigh, int len, const Ipp16s* pTabs, int tapsLen, int
   scaleFactor);
IppStatus ippsLowHighFilter_Aurora_32f(const Ipp32f* pSrc, Ipp32f* pDstLow,
   Ipp32f* pDstHigh, int len, const Ipp32f* pTabs, int tapsLen);
```

HighBandCoding_Aurora

Codes and decodes the high frequency band energy values.

```
IppStatus ippsHighBandCoding_Aurora_32f(const Ipp32f* pSrcHFB, const Ipp32f*
   pInSWP, const Ipp32f* pDSWP, Ipp32f* pDstHFB);
IppStatus ippsHighBandCoding_Aurora_32s_Sfs(const Ipp32s* pSrcHFB, const
   Ipp32s* pInSWP, const Ipp32s* pDSWP, Ipp32s* pDstHFB, int scaleFactor)
```

BlindEqualization_Aurora

Equalizes the cepstral coefficients.

```
IppStatus ippsBlindEqualization_Aurora_32f(const Ipp32f* pRefs, Ipp32f*
   pCeps, Ipp32f* pBias, int len, Ipp32f val);
IppStatus ippsBlindEqualization_Aurora_16s(const Ipp16s* pRefsQ6, Ipp16s*
   pCeps, Ipp16s* pBias, int len, Ipp32s valQ6);
```

DeltaDelta_Aurora
Calculates the first and second derivatives according to ETSI ES 202 050 standard.

```
IppStatus ippsDeltaDelta_Aurora_16s_D2Sfs(const Ipp16s* pSrc, Ipp16s*
   pDst, int dstStep, int height, int deltaMode, int scaleFactor);
IppStatus ippsDeltaDeltaMul_Aurora_16s_D2Sfs(const Ipp16s* pSrc, const
   Ipp16s* pVal, Ipp16s* pDst, int dstStep, int height, int deltaMode,
   int scaleFactor);
IppStatus ippsDeltaDelta_Aurora_32f_D2(const Ipp32f* pSrc, Ipp32f* pDst,
   int dstStep, int height, int deltaMode);
IppStatus ippsDeltaDeltaMul_Aurora_32f_D2(const Ipp32f* pSrc, const
   Ipp32f* pVal, Ipp32f* pDst, int dstStep, int height, int deltaMode);
```

VADGetBufSize_Aurora
Queries the memory size for VAD decision.

```
IppStatus ippsVADGetBufSize_Aurora_32f(int* pSize);
IppStatus ippsVADGetBufSize_Aurora_16s(int* pSize);
```

VADInit_Aurora
Gets the VAD structure size.

```
IppStatus ippsVADInit_Aurora_32f(char* pVADmem);
IppStatus ippsVADInit_Aurora_16s(char* pVADmem);
```

VADDecision_Aurora
Takes the VAD decision.

```
IppStatus ippsVADDecision_Aurora_32f(const Ipp32f* pCoeff, const Ipp32f*
   pTrans, IppVADDecision_Aurora*  pRes, int nbSpeechFrame, char*
   pVADmem);
IppStatus ippsVADDecision_Aurora_16s(const Ipp16s* pCoeff, const Ipp16s*
   pTrans, IppVADDecision_Aurora*  pRes,int nbSpeechFrame, char*
   pVADmem);
```

VADFlush_Aurora
Takes VAD decision for zero input frame.

```
IppStatus ippsVADFlush_Aurora_32f(IppVADDecision_Aurora* pRes, char*
   pVADmem);
IppStatus ippsVADFlush_Aurora_16s(IppVADDecision_Aurora* pRes, char*
   pVADmem);
```

Ephraim-Malah Noise Suppressor

FilterUpdateEMNS
Calculates the noise suppression filter coefficients.

```
IppStatus ippsFilterUpdateEMNS_32s(const Ipp32s* pSrcWienerCoefsQ31,
   const Ipp32s* pSrcPostSNRQ15, Ipp32s* pDstFilterCoefsQ31, int len);
```

```
IppStatus ippsFilterUpdateEMNS_32f(const Ipp32f* pSrcWienerCoefs, const
   Ipp32f* pSrcPostSNR, Ipp32f* pDstFilterCoefs, int len);
```

FilterUpdateWiener

Calculates the Wiener filter coefficients.

```
IppStatus ippsFilterUpdateWiener_32s(const Ipp32s*
   pSrcPriorSNRQ15,Ipp32s* pDstFilterCoefsQ31, int len);
IppStatus ippsFilterUpdateWiener_32f(const Ipp32f* pSrcPriorSNR, Ipp32f*
   pDstFilterCoefs, int len);
```

GetSizeMCRA

Calculates the size in bytes required for the `IppMCRAState` state structure.

```
IppStatus ippsGetSizeMCRA_32s(int nFFTSize, int* pDstSize);
IppStatus ippsGetSizeMCRA_32f(int nFFTSize, int* pDstSize);
```

InitMCRA

Initializes the IppMCRAState state structure.

```
IppStatus ippsInitMCRA_32s(int nSamplesPerSec, int nFFTSize,
   IppMCRAState* pDst);
IppStatus ippsInitMCRA_32f(int nSamplesPerSec, int nFFTSize,
   IppMCRAState32f* pDst);
```

AltInitMCRA

Allocates memory and initializes the IppMCRAState state structure.

```
IppStatus ippsAltInitMCRA_32s(int nSamplesPerSec, int nFFTSize,
   IppMCRAState** ppDst);
IppStatus ippsAltInitMCRA_32f(int nSamplesPerSec, int nFFTSize, int
   nUpdateSamples, IppMCRAState32f* pDst);
```

UpdateNoisePSDMCRA

Re-estimates the noise power spectrum.

```
IppStatus ippsUpdateNoisePSDMCRA_32s_I(const Ipp32s* pSrcNoisySpeech,
   IppMCRAState* pSrcDstState, Ipp32s* pSrcDstNoisePSD);
IppStatus ippsUpdateNoisePSDMCRA_32f_I(const Ipp32f* pSrcNoisySpeech,
   IppMCRAState32f* pSrcDstState, Ipp32f* pSrcDstNoisePSD);
```

Acoustic Echo Canceller

FilterAECNLMS

Computes the frequency-domain adaptive filter output.

```
IppStatus ippsFilterAECNLMS_32sc_Sfs(Ipp32sc** ppSrcSignalIn,  Ipp32sc**
   ppSrcCoefs, Ipp32sc* pDstSignalOut, int numSegments,  int len, int
   scaleFactor);
```

CoefUpdateAECNLMS
Updates the adaptive filter coefficients.

```
IppStatus ippsCoefUpdateAECNLMS_32sc_I(const IppAECScaled32s*
   pSrcStepSize, const Ipp32sc** ppSrcFilterInput, const Ipp32sc*
   pSrcError, Ipp32sc** ppSrcDstCoefs, int numSegments, int len, int
   scaleFactorCoef);
```

StepSizeUpdateAECNLMS
Computes the adaptive step size.

```
IppStatus ippsStepSizeUpdateAECNLMS_32s(const Ipp32s* pSrcInputPSD,
   Ipp32s muQ31, IppAECScaled32s maxStepSize, Ipp32s minInputPSD,
   IppAECScaled32s* pDstStepSize, int len);
```

ControllerGetSizeAEC
Returns the size of the AEC controller state structure.

```
IppStatus ippsControllerGetSizeAEC_32s(int* pDstSize);
```

ControllerInitAEC
Initializes the AEC controller state structure.

```
IppStatus ippsControllerInitAEC_32s(const IppAECNLMSParam* pSrcParams,
   IppAECCtrlState* pDstState);
```

ControllerUpdateAEC
Implements the energy-based AEC controller.

```
IppStatus ippsControllerUpdateAEC_32s(const IppAECNLMSParam* pSrcParams,
   IppAECCtrlState* pSrcDstState, Ipp32s* pDstMuQ31,  Ipp32s*
   pDstAECOutGainQ30, Ipp32s* pDstSpeakerGainQ30);
```

Voice Activity Detector

FindPeaks
Identifies peaks in the input vector.

```
IppStatus ippsFindPeaks_32s8u(const Ipp32s* pSrc, Ipp8u* pDstPeaks, int
   len, int searchSize, int movingAvgSize);

IppStatus ippsFindPeaks_32f8u(const Ipp32f* pSrc,  Ipp8u* pDstPeaks, int
   len, int searchSize, int movingAvgSize);
```

PeriodicityLSPE
Computes the periodicity of the input speech frame.

```
IppStatus ippsPeriodicityLSPE_16s(const Ipp16s* pSrc, int len, Ipp16s*
   pPeriodicityQ15, int* period, int maxPeriod, int minPeriod);

IppStatus ippsPeriodicityLSPE_32f(const Ipp32f* pSrc, int len, Ipp32f*
   pPeriodicity, int* period, int maxPeriod, int minPeriod);
```

Periodicity
Computes the periodicity of the input block.

```
IppStatus ippsPeriodicity_32s16s(const Ipp32s* pSrc, int len, Ipp16s*
   pPeriodicityQ15, int* period, int maxPeriod, int minPeriod);
IppStatus ippsPeriodicity_32f (const Ipp32f* pSrc,  int len, Ipp32f*
   pPeriodicity, int* period, int maxPeriod, int minPeriod);
```

Speech Coding Functions

ConvPartial
Performs linear convolution of 1D signals.

```
IppStatus ippsConvPartial_16s_Sfs (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsConvPartial_16s32s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, ipp32s* pDst, int len);
IppStatus ippsConvPartial_NR_16s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pDst, int len);
```

InterpolateC_NR
Computes the weighted sum of two vectors

```
IppStatus ippsInterpolateC_NR_16s(const Ipp16s* pSrc1, Ipp16s val1, int
   val1ScaleFactor, const Ipp16s* pSrc2, Ipp16s val2, int
   val2ScaleFactor, Ipp16s* pDst, int len);
```

Mul_NR
Multiplies the elements of two vectors.

```
IppStatus ippsMul_NR_16s_Sfs (const Ipp16s* pSrc1, const Ipp16s* pSrc2,
   Ipp16s* pDst, int len, int scaleFactor);
IppStatus ippsMul_NR_16s_ISfs (const Ipp16s* pSrc, Ipp16s* pSrcDst, int
   len, int scaleFactor);
```

MulC_NR
Multiplies each element of a vector by a constant value.

```
IppStatus ippsMulC_NR_16s_Sfs (Ipp16s* pSrc, Ipp16s val, Ipp16s* pDst,
   int len, int scaleFactor);
IppStatus ippsMulC_NR_16s_ISfs (Ipp16s val, Ipp16s* pSrcDst, int len, int
   scaleFactor);
```

MulPowerC_NR
Performs power weighting for each element of a vector.

```
IppStatus ippsMulPowerC_NR_16s_Sfs (const Ipp16s* pSrc, Ipp16s val,
   Ipp16s* pDst, int len, int scaleFactor);
```

AutoScale
Scales by the maximal elements.

```
IppStatus ippsAutoScale_16s (const Ipp16s* pSrc, Ipp16s* pDst, int len,
   int* pScale);
IppStatus ippsAutoScale_16s_I(Ipp16s* pSrcDst, int len, int* pScale);
```

DotProdAutoScale
Computes the dot product of two vectors using the automatic scaling.

```
IppStatus ippsDotProdAutoScale_16s32s_Sfs(const Ipp16s* pSrc1, const
   Ipp16s* pSrc2, int len, Ipp32s* pDp, int* pSfs);
```

InvSqrt
Computes inverse square root of vector elements.

```
IppStatus ippsInvSqrt_32s_I (Ipp32s* pSrcDst, int len );
```

AutoCorr
Calculates autocorrelation of a vector.

```
IppStatus ippsAutoCorr_16s32s (const Ipp16s* pSrc, int srcLen, Ipp32s*
   pDst, int dstLen );
```

AutoCorrLagMax
Estimates the maximum auto-correlation of a vector.

```
IppStatus ippsAutoCorrLagMax_Inv_16s (const Ipp16s* pSrc, int len, int
   lowerLag, int upperLag, Ipp32s* pMax, int* pMaxLag);
IppStatus ippsAutoCorrLagMax_Fwd_16s (const Ipp16s* pSrc, int len, int
   lowerLag, int upperLag, Ipp32s* pMax, int* pMaxLag);
IppStatus ippsAutoCorrLagMax_32f (const Ipp32f* pSrc, int len, int
   lowerLag, int upperLag, Ipp32f* pMax, int* pMaxLag);
```

AutoCorr_NormE
Estimates normal auto-correlation of a vector.

```
IppStatus ippsAutoCorr_NormE_16s32s (const Ipp16s* pSrc, int len, Ipp32s*
   pDst, int lenDst, int* pNorm);
IppStatus ippsAutoCorr_NormE_NR_16s (const Ipp16s* pSrc, int len, Ipp16s*
   pDst, int lenDst, int* pNorm);
```

CrossCorr
Estimates the cross-correlation of two vectors.

```
IppStatus ippsCrossCorr_16s32s_Sfs (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pDst, int scaleFactor);
IppStatus ippsCrossCorr_NormM_16s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp16s* pDst);
```

CrossCorrLagMax
Estimates the maximum cross-correlation between two vectors.

```
IppStatus ippsCrossCorrLagMax_16s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, int lag, Ipp32s* pMax, int* pMaxLag);
```

```
IppStatus ippsCrossCorrLagMax_32f64f(const Ipp32f* pSrc1, const Ipp32f*
   pSrc2, int len, int lag, Ipp64f* pMax, int* pMaxLag);
```

SynthesisFilter

Computes the speech signal by filtering the input speech through the synthesis filter 1/A(z).

```
IppStatus ippsSynthesisFilter_NR_16s_Sfs(const Ipp16s*  pLPC, const
   Ipp16s*  pSrc, Ipp16s*  pDst, int len, int scaleFactor, const Ipp16s*
   pMem);
IppStatus ippsSynthesisFilterLow_NR_16s_ISfs(const Ipp16s*  pLPC,
   Ipp16s*  pSrcDst, int len, int scaleFactor, const Ipp16s* pMem);
IppStatus ippsSynthesisFilter_NR_16s_ISfs(const Ipp16s*  pLPC, Ipp16s*
   pSrcDst, int len, int scaleFactor, const Ipp16s* pMem);
```

G.729 Related Functions

DotProd_G729

Computes the dot product of two vectors.

```
IppStatus ippsDotProd_G729A_16s32s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, int len, Ipp32s* pDp);
IppStatus ippsDotProd_G729A_32f(const Ipp32f* pSrc1, const Ipp32f*
   pSrc2, int len, Ipp32f* pDp);
```

Interpolate_G729

Computes the weighted sum of two vectors.

```
IppStatus ippsInterpolate_G729_16s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pDst, int len);
IppStatus ippsInterpolateC_G729_16s_Sfs (const Ipp16s* pSrc1, Ipp16s
   val1, const Ipp16s* pSrc2, Ipp16s val2, Ipp16s* pDst, int len, int
   scaleFactor);
IppStatus ippsInterpolateC_NR_G729_16s_Sfs (const Ipp16s* pSrc1, Ipp16s
   val1, const Ipp16s* pSrc2, Ipp16s val2, Ipp16s* pDst, int len, int
   scaleFactor);
IppStatus ippsInterpolateC_G729_32f(const Ipp32f* pSrc1, Ipp32f val1,
   const Ipp32f* pSrc2, Ipp32f val2, Ipp32f* pDst, int len);
```

AutoCorr_G729

Estimates the auto-correlation of a vector.

```
IppStatus ippsAutoCorr_G729B(const Ipp16s*  pSrcSpch,
   Ipp16s*  pResultAutoCorrExp, Ipp32s*  pDstAutoCorr);
```

LevinsonDurbin_G729

Calculates LP coefficients from the autocorrelation coefficients.

```
IppStatus ippsLevinsonDurbin_G729_32s16s(const Ipp32s* pSrcAutoCorr, int
   order, Ipp16s* pDstLPC, Ipp16s* pDstRc, Ipp16s*
   pResultResidualEnergy);
```

```
IppStatus ippsLevinsonDurbin_G729_32f(const Ipp32f* pSrcAutoCorr, int
   order, Ipp32f* pDstLpc, Ipp32f* pDstRc, Ipp32f*
   pResultResidualEnergy);
IppStatus ippsLevinsonDurbin_G729B(const Ipp32s*  pSrcAutoCorr, Ipp16s*
   pDstLPC, Ipp16s*  pDstRC, Ipp16s*  pResultResidualEnergy);
```

LPCToLSP_G729

Converts LP coefficients to LSP coefficients.

```
IppStatus ippsLPCToLSP_G729_16s(const Ipp16s*  pSrcLPC, const Ipp16s*
   pSrcPrevLSP, Ipp16s*  pDstLSP);
IppStatus ippsLPCToLSP_G729A_16s(const Ipp16s*  pSrcLPC, const Ipp16s*
   pSrcPrevLSP, Ipp16s*  pDstLSP);
IppStatus ippsLPCToLSP_G729A_32f(const Ipp32f* pSrcLPC, const Ipp32f*
   pSrcPrevLsp, Ipp32f* pDstLSP);
IppStatus ippsLPCToLSP_G729_32f(const Ipp32f* pSrcLPC, const Ipp32f*
   pSrcPrevLsp, Ipp32f* pDstLSP);
```

LSFToLSP_G729

Converts Line Spectral Frequencies to LSP coefficients.

```
IppStatus ippsLSFToLSP_G729_16s (const Ipp16s* pLSF, Ipp16s* pLSP);
```

LSFQuant_G729

Performs quantization of LSF coefficients.

```
IppStatus ippsLSFQuant_G729_16s (const Ipp16s* pLSF, Ipp16s*
   pQuantLSFTable, Ipp16s* pQuantLSF, Ipp16s* quantIndex);
IppStatus ippsLSFQuant_G729B_16s (const Ipp16s* pLSF, Ipp16s*
   pQuantLSFTable, Ipp16s* pQuantLSF, Ipp16s* quantIndex);
IppStatus ippsLSFQuant_G729B_32f (const Ipp32f* pLSF, Ipp32f*
   pQuantLSFTable, Ipp32f* pQuantLSF,  int*  quantIndex);
```

LSFDecode_G729

Decodes quantized LSFs.

```
IppStatus ippsLSFDecode_G729_16s (const Ipp16s* quantIndex, Ipp16s*
   pQuantLSFTable, Ipp16s* pQuantLSF);
IppStatus ippsLSFDecode_G729B_16s(const Ipp16s* quantIndex, Ipp16s*
   pQuantLSFTable, Ipp16s*  pQuantLSF);
IppStatus ippsLSFDecode_G729B_32f (const int* quantIndex, Ipp32f*
   pQuantLSFTable, Ipp32f* pDstQLsp);
IppStatus ippsLSFDecode_G729_32f (const int* quantIndex, Ipp32f*
   pQuantLSFTable, Ipp32f* pQuantLSF);
```

LSFDecodeErased_G729

Reconstructs quantized LSFs in case when a frame is erased.

```
IppStatus ippsLSFDecodeErased_G729_16s (Ipp16s maIndex, Ipp16s*
   pQuantLSFTable, Ipp16s* pQuantLSF);
```

```
IppStatus ippsLSFDecodeErased_G729_32f (int maIndex, Ipp32f*
   pQuantLSFTable,        const Ipp32f* pSrcPrevLSF);
```

LSPToLPC_G729

Converts LSP coefficients to LP coefficients.

```
IppStatus ippsLSPToLPC_G729_16s(const Ipp16s*  pSrcLSP, Ipp16s*
   pDstLPC);
IppStatus ippsLSPToLPC_G729_32f(const Ipp32f* pSrcLsp, Ipp32f* pDstLpc);
```

LSPQuant_G729

Quantizes the LSP coefficients.

```
IppStatus ippsLSPQuant_G729_16s(const Ipp16s*  pSrcLSP, Ipp16s*
   pSrcDstPrevFreq, Ipp16s*  pDstQLSP, Ipp16s*  pDstQLSPIndex);
IppStatus ippsLSPQuant_G729E_32f (const Ipp32f*  pSrcLSP,  Ipp32f*
   pSrcDstPrevFreq, Ipp32f* pDstQLSF, Ipp32f*  pDstQLSP, int*
   pDstQLspIndex);
```

LSPToLSF_G729

Converts LSP coefficients to LSF coefficients.

```
IppStatus ippsLSPToLSF_Norm_G729_16s (const Ipp16s* pLSP, Ipp16s* pLSF);
IppStatus ippsLSPToLSF_G729_16s (const Ipp16s* pLSP, Ipp16s* pLSF);
```

LagWindow_G729

Applies 60Hz bandwidth expansion.

```
IppStatus ippsLagWindow_G729_32s_I (Ipp32s* pSrcDst, int len);
```

OpenLoopPitchSearch_G729

Searches for an optimal pitch value.

```
IppStatus ippsOpenLoopPitchSearch_G729_16s(const Ipp16s*  pSrc, Ipp16s*
   bestLag);
IppStatus ippsOpenLoopPitchSearch_G729A_16s(const Ipp16s*  pSrc, Ipp16s*
   bestLag);
IppStatus ippsOpenLoopPitchSearch_G729A_32f(const Ipp32f* pSrc, Ipp16s*
   bestLag);
```

AdaptiveCodebookSearch_G729

Searches for the integer delay and the fraction delay, and computes the adaptive vector.

```
IppStatus ippsAdaptiveCodebookSearch_G729_16s(Ipp16s valOpenDelay, const
   Ipp16s*  pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse, Ipp16s*
   pSrcDstPrevExcitation, Ipp16s*  pDstDelay, Ipp16s*  pDstAdptVector,
   Ipp16s subFrame);
IppStatus ippsAdaptiveCodebookSearch_G729A_16s(Ipp16s valOpenDelay,
   const Ipp16s*  pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse,
   Ipp16s*  pSrcDstPrevExcitation, Ipp16s*  pDstDelay, Ipp16s*
   pDstAdptVector, Ipp16s subFrame);
```

```
IppStatus ippsAdaptiveCodebookSearch_G729D_16s  (Ipp16s valOpenDelay,
   const Ipp16s* pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse,
   Ipp16s* pSrcDstPrevExcitation, Ipp16s*  pDstDelay, Ipp16s subFrame);
```

DecodeAdaptiveVector_G729

Restores the adaptive codebook vector by interpolating the past excitation.

```
IppStatus ippsDecodeAdaptiveVector_G729_16s(const Ipp16s* pSrcDelay,
   Ipp16s* pSrcDstPrevExcitation, Ipp16s*  pDstAdptVector);

IppStatus ippsDecodeAdaptiveVector_G729_16s_I(const Ipp16s* pSrcDelay,
   Ipp16s* pSrcDstPrevExcitation);

IppStatus ippsDecodeAdaptiveVector_G729_32f_I(const Ipp32s* pSrcDelay,
   Ipp32f* pSrcDstPrevExcitation);
```

FixedCodebookSearch_G729

Searches for the fixed codebook vector.

```
IppStatus ippsFixedCodebookSearch_G729_16s(const Ipp16s*  pSrcFixedCorr,
   Ipp16s*  pSrcDstMatrix, Ipp16s*  pDstFixedVector, Ipp16s*
   pDstFixedIndex, Ipp16s*  pSearchTimes, Ipp16s subFrame);

IppStatus ippsFixedCodebookSearch_G729_32s16s(const Ipp16s*
   pSrcFixedCorr, Ipp32s*  pSrcDstMatrix, Ipp16s*  pDstFixedVector,
   Ipp16s*  pDstFixedIndex, Ipp16s*  pSearchTimes, Ipp16s subFrame);

IppStatus ippsFixedCodebookSearch_G729A_16s(const Ipp16s*
   pSrcFixedCorr,  Ipp16s*  pSrcDstMatrix, Ipp16s*  pDstFixedVector,
   Ipp16s*  pDstFixedIndex);

IppStatus ippsFixedCodebookSearch_G729A_32s16s(const Ipp16s*
   pSrcFixedCorr, Ipp32s*  pSrcDstMatrix, Ipp16s*  pDstFixedVector,
   Ipp16s*  pDstFixedIndex);

IppStatus ippsFixedCodebookSearch_G729E_16s(int mode, const Ipp16s*
   pSrcFixedTarget, const Ipp16s* pSrcLtpResidual, const Ipp16s*
   pSrcImpulseResponse, Ipp16s* pDstFixedVector, Ipp16s*
   pDstFltFixedVector, Ipp16s* pDstFixedIndex);

IppStatus ippsFixedCodebookSearch_G729D_16s(const Ipp16s* pSrcFixedCorr,
   const Ipp16s* pSrcImpulseResponse, Ipp16s* pSrcDstMatrix, Ipp16s*
   pDstFixedVector, Ipp16s* pDstFltFixedVector, Ipp16s* pDstFixedIndex);

IppStatus ippsFixedCodebookSearch_G729_32f (const Ipp32f* pSrcFixedCorr,
   Ipp32f* pSrcDstMatrix, Ipp32f* pDstFixedVector, Ipp32s*
   pDstFixedIndex, Ipp32s* pSearchTimes, Ipp32s subFrame);

IppStatus ippsFixedCodebookSearch_G729A_32f (const Ipp32f*
   pSrcFixedCorr, Ipp32f* pSrcDstMatrix, Ipp32f* pDstFixedVector,
   Ipp32s* pDstFixedIndex);

IppStatus ippsFixedCodebookSearch_G729D_32f (Ipp32f* pSrcDstFixedCorr,
   Ipp32f* pSrcDstMatrix, const Ipp32f* pSrcImpulseResponse, Ipp32f*
   pDstFixedVector, Ipp32f* pDstFltFixedVector, Ipp32s* pDstFixedIndex);
```

```
IppStatus ippsFixedCodebookSearch_G729E_32f (int mode, Ipp32f*
   pSrcDstFixedCorr, const Ipp32f* pSrcLtpResidual, const Ipp32f*
   pSrcImpulseResponse, Ipp32f* pDstFixedVector, Ipp32f*
   pDstFltFixedVector, Ipp32s* pDstFixedIndex);
```

GainCodebookSearch

Finds pitch and fixed gains.

```
IppStatus ippsGainCodebookSearch_G729_32f( const Ipp32f*
   pSrcCorrFactors, Ipp32f valPredictedCodebookGain, const int*
   pSrcCand, Ipp32f*  pDstIdxs, Ipp32s valTimeFlag);

IppStatus ippsGainCodebookSearch_G729D_32f (const Ipp32f*
   pSrcCorrFactors, Ipp32f valPredictedCodebookGain, const int*
   pSrcCand, Ipp32f*  pDstIdxs, Ipp32s valTimeFlag);
```

ToeplizMatrix_G729

Calculates elements of the Toepliz matrix for the fixed codebook search.

```
IppStatus ippsToeplizMatrix_G729_16s(const Ipp16s*  pSrcImpulseResponse,
   Ipp16s*  pDstMatrix);

IppStatus ippsToeplizMatrix_G729_16s32s(const Ipp16s*
   pSrcImpulseResponse, Ipp32s*  pDstMatrix);

IppStatus ippsToeplizMatrix_G729_32f(const Ipp32f*
   pSrcImpulseResponse,, Ipp32f*  pDstMatrix);

IppStatus ippsToeplizMatrix_G729D_32f(const Ipp32f*
   pSrcImpulseResponse,, Ipp32f*  pDstMatrix);
```

DecodeGain_G729

Decodes the adaptive and fixed-codebook gains.

```
IppStatus ippsDecodeGain_G729_16s (Ipp32s energy, Ipp16s* pPastEnergy,
   const Ipp16s* pQuaIndex, Ipp16s* pGain);

IppStatus ippsDecodeGain_G729I_16s(Ipp32s energy,  Ipp16s
   valGainAttenuation, Ipp16s* pPastEnergy, const Ipp16s* pQuaIndex,
   Ipp16s* pGain);
```

GainControl_G729

Calculates adaptive gain control.

```
IppStatus ippsGainControl_G729_16s_I (const Ipp16s* pSrc, Ipp16s*
   pSrcDst, Ipp16s* pGain);

IppStatus ippsGainControl_G729A_16s_I (const Ipp16s* pSrc, Ipp16s*
   pSrcDst, Ipp16s* pGain);

IppStatus ippsGainControl_G729_32f_I (Ipp32f gainScalingFactor, Ipp32f
   gainFactor, Ipp32f* pSrcDst, Ipp32f* pGain);
```

GainQuant_G729

Quantizes the codebook gain using a two-stage conjugate-structured codebook.

```
IppStatus ippsGainQuant_G729_16s(const Ipp16s*  pSrcAdptTarget, const
    Ipp16s*  pSrcFltAdptVector, const Ipp16s*  pSrcFixedVector, const
    Ipp16s* pSrcFltFixedVector, Ipp16s* pSrcDstEnergyErr, Ipp16s*
    pDstQGain, Ipp16s*  pDstQGainIndex, Ipp16s tameProcess);

IppStatus ippsGainQuant_G729D_16s(const Ipp16s* pSrcAdptTarget, const
    Ipp16s* pSrcFltAdptVector, const Ipp16s* pSrcFixedVector, const
    Ipp16s*  pSrcFltFixedVector, Ipp16s* pSrcDstEnergyErr, Ipp16s*
    pDstQGain, Ipp16s* pDstQGainIndex, Ipp16s tameProcess)
```

AdaptiveCodebookContribution_G729

Updates target vector for codebook search by subtracting adaptive codebook contribution.

```
IppStatus ippsAdaptiveCodebookContribution_G729_16s (Ipp16s gain, const
    Ipp16s* pFltAdptVector, const Ipp16s* pSrcAdptTarget, Ipp16s*
    pDstAdptTarget);

IppStatus ippsAdaptiveCodebookContribution_G729_32f (Ipp32f gain, const
    Ipp32f* pFltAdptVector, const Ipp32f* pSrcAdptTarget, Ipp32f*
    pDstAdptTarget);
```

AdaptiveCodebookGain_G729

Calculates the gain of the adaptive-codebook vector and the filtered codebook vector.

```
IppStatus ippsAdaptiveCodebookGain_G729_16s(const Ipp16s*
    pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse, const Ipp16s*
    pSrcAdptVector, Ipp16s*  pDstFltAdptVector, Ipp16s*
    pResultAdptGain);

IppStatus ippsAdaptiveCodebookGain_G729A_16s(const Ipp16s*
    pSrcAdptTarget, const Ipp16s*  pSrcLPC, const Ipp16s*
    pSrcAdptVector, Ipp16s*  pDstFltAdptVector, Ipp16s*
    pResultAdptGain);
```

ResidualFilter_G729

Implements an inverse LP filter and obtains the residual signal.

```
IppStatus ippsResidualFilter_G729_16s(const Ipp16s*  pSrcSpch, const
    Ipp16s*  pSrcLPC, Ipp16s*  pDstResidual);

IppStatus ippsResidualFilter_G729E_16s(const Ipp16s* pCoeffs, int order,
    const Ipp16s* pSrc, Ipp16s* pDst, int len)
```

SynthesisFilter_G729

Reconstructs the speech signal from LP coefficients and residuals.

```
IppStatus ippsSynthesisFilter_G729_16s(const Ipp16s*  pSrcResidual,
    const Ipp16s*  pSrcLPC, Ipp16s*  pSrcDstSpch);

IppStatus ippsSynthesisFilterZeroStateResponse_NR_16s(const Ipp16s*
    pSrcLPC, Ipp16s* pDstImp, int len, int scaleFactor);

IppStatus ippsSynthesisFilter_G729E_16s(const Ipp16s* pLPC, int order,
    const Ipp16s* pSrc, Ipp16s* pDst, int len, const Ipp16s* pMem);
```

```
IppStatus ippsSynthesisFilter_G729E_16s_I(const Ipp16s* pLPC, int order,
   Ipp16s* pSrcDst, int len, const Ipp16s* pMem);
IppStatus ippsSynthesisFilter_G729_32f(const Ipp32f* pLpc, int order,
   const Ipp32f* pSrc, Ipp32f* pDst, int len, Ipp32f* pMem);
```

LongTermPostFilter_G729

Restores the long-term information from the old speech signal.

```
IppStatus ippsLongTermPostFilter_G729_16s (Ipp16s gammaFactor, int
   valDelay, const Ipp16s* pSrcDstResidual, Ipp16s* pDstFltResidual,
   Ipp16s* pResultVoice);
IppStatus ippsLongTermPostFilter_G729A_16s(Ipp16s valDelay, const
   Ipp16s*  pSrcSpch, const Ipp16s*  pSrcLPC, Ipp16s*  pSrcDstResidual,
   Ipp16s*  pDstFltResidual);
IppStatus ippsLongTermPostFilter_G729B_16s(Ipp16s valDelay, const
   Ipp16s*  pSrcSpch, const Ipp16s*  pSrcLPC, Ipp16s*  pSrcDstResidual,
   Ipp16s*  pDstFltResidual, Ipp16s*  pResultVoice, Ipp16s frameType);
```

ShortTermPostFilter_G729

Restores speech signal from the residuals.

```
IppStatus ippsShortTermPostFilter_G729_16s(const Ipp16s*  pSrcLPC, const
   Ipp16s*  pSrcFltResidual, Ipp16s*  pSrcDstSpch, Ipp16s*
   pDstImpulseResponse);
IppStatus ippsShortTermPostFilter_G729A_16s(const Ipp16s*  pSrcLPC,
   const Ipp16s*  pSrcFltResidual, Ipp16s*  pSrcDstSpch);
```

TiltCompensation_G729

Compensates for the tilt in the short-term filter.

```
IppStatus ippsTiltCompensation_G729_16s(const Ipp16s*
   pSrcImpulseResponse, Ipp16s*  pSrcDstSpch);
IppStatus ippsTiltCompensation_G729A_16s(const Ipp16s*  pSrcLPC, Ipp16s*
   pSrcDstFltResidual);
IppStatus ippsTiltCompensation_G729E_16s(Ipp16s val, const Ipp16s* pSrc,
   Ipp16s* pDst);
```

HarmonicFilter

Calculates the harmonic filter.

```
IppStatus ippsHarmonicFilter_16s_I (Ipp16s beta, int T, Ipp16s* pSrcDst,
   int len);
IppStatus ippsHarmonicFilter_NR_16s (Ipp16s beta, int T, const Ipp16s*
   pSrc, Ipp16s*  pDst, int len);
IppStatus ippsHarmonicFilter_32f_I(Ipp32f beta, int T, Ipp32f* pSrcDst,
   int len);
```

HighPassFilterSize_G729

Returns the G729 high-pass filter size.

```
IppStatus ippsHighPassFilterSize_G729 (int* pSize);
```

HighPassFilterInit_G729
Initializes the high-pass filter.

```
IppStatus ippsHighPassFilterInit_G729 (Ipp16s* pCoeff, char*
   pMemUpdated);
```

HighPassFilter_G729
Performs G729 high-pass filtering.

```
IppStatus ippsHighPassFilter_G729_16s_ISfs (Ipp16s* pSrcDst, int len,
   int scaleFactor, char* pMemUpdated);
```

IIR16s_G729
Performs IIR filtering.

```
IppStatus ippsIIR16sLow_G729_16s(const Ipp16s* pCoeffs, const Ipp16s*
   pSrc, Ipp16s* pDst, Ipp16s* pMem);
IppStatus ippsIIR16s_G729_16s(const Ipp16s* pCoeffs, const Ipp16s* pSrc,
   Ipp16s* pDst, Ipp16s* pMem);
```

PhaseDispersionGetStateSize_G729D
Queries the memory length of the phase dispersion filter.

```
IppStatus ippsPhaseDispersionGetStateSize_G729D_16s (int* pSize);
```

PhaseDispersionInit_G729D
Initializes the phase dispersion filter memory.

```
IppStatus ippsPhaseDispersionInit_G729D_16s
   (IppsPhaseDispersion16s_State_G729D* pPhDMem);
```

PhaseDispersionUpdate_G729D
Updates the phase dispersion filter state.

```
IppStatus ippsPhaseDispersionUpdate_G729D_16s(Ipp16s valPitchGain,
   Ipp16s valCodebookGain, IppsPhaseDispersion16s_State_G729D* pPhDMem);
```

PhaseDispersion_G729D
Performs the phase dispersion filtering.

```
IppStatus ippsPhaseDispersion_G729D_16s(Ipp16s valCodebookGain, Ipp16s
   valPitchGain,  const Ipp16s* pSrcExcSignal, Ipp16s* pDstFltExcSignal,
   Ipp16s* pSrcDstInnovation, IppsPhaseDispersion16s_State_G729D*
   pPhDMem)
```

Preemphasize_G729A
Computes pre-emphasis of a post filter.

```
IppStatus ippsPreemphasize_G729A_16s (Ipp16s gamma, const Ipp16s* pSrc,
   Ipp16s* pDst, int len, Ipp16s* pMem);
IppStatus ippsPreemphasize_G729A_16s_I (Ipp16s gamma, Ipp16s* pSrcDst,
   int len, Ipp16s* pMem);
IppStatus ippsPreemphasize_32f_I(Ipp32f gamma,  Ipp32f* pSrcDst,  int
   len, Ipp32f* pMem);
```

WinHybridGetStateSize_G729E
Queries the length of the hybrid windowing module memory.

```
IppStatus ippsWinHybridGetStateSize_G729E_16s (int* pSize);
IppStatus ippsWinHybridGetStateSize_G729E_32f (int* pSize);
```

WinHybridInit_G729E
Initializes the hybrid windowing module memory.

```
IppStatus ippsWinHybridInit_G729E_16s (IppsWinHybridState_G729E_16s*
   pMem);
IppStatus ippsWinHybridInit_G729E_32f(IppsWinHybridState_G729E_32f*
   pMem);
```

WinHybrid_G729E
Applies the hybrid window and computes autocorrelation coefficients.

```
IppStatus ippsWinHybrid_G729E_16s32s (const Ipp16s* pSrcSynthSpeech,
   Ipp32s* pDstInvAutoCorr, IppsWinHybridState_G729E_16s* pMem);
IppStatus ippsWinHybrid_G729E_32f(const Ipp32f* pSrcSynthSpeech, Ipp32f*
   pDstInvAutoCorr, IppsWinHybridState_G729E_32f* pMem);
```

RandomNoiseExcitation_G729B
Initializes a random vector with a Gaussian distribution.

```
IppStatus ippsRandomNoiseExcitation_G729B_16s (Ipp16s* pSeed, Ipp16s*
   pExc, int len);
IppStatus ippsRandomNoiseExcitation_G729B_16s32f(Ipp16s* pSeed, Ipp32f*
   pExc, int len);
```

FilteredExcitation_G729
Computes filtered exitation.

```
IppStatus ippsFilteredExcitation_G729_32f(const Ipp32f*
   pSrcImpulseResponse, Ipp32f* pSrcDstPastExc, Ipp32f valExc, Ipp32s
   len);
```

G.723.1 Related Functions

AutoCorr_G723
Estimates the auto-correlation of a vector.

```
IppStatus ippsAutoCorr_G723_16s(const Ipp16s* pSrcSpch,  Ipp16s*
   pResultAutoCorrExp, Ipp16s* pDstAutoCorr);
```

AutoCorr_NormE_G723
Estimates normal auto-correlation of a vector.

```
IppStatus ippsAutoCorr_NormE_G723_16s(const Ipp16s* pSrc, Ipp16s* pDst,
   int* pNorm);
```

LevinsonDurbin_G723

Calculates the LP coefficients from the autocorrelation coefficients.

```
IppStatus ippsLevinsonDurbin_G723_16s(const Ipp16s*  pSrcAutoCorr,
   Ipp16s* pValResultSineDtct, Ipp16s* pResultResidualEnergy, Ipp16s*
   pDstLPC);
```

LPCToLSF_G723

Converts LP coefficients to LSF coefficients.

```
IppStatus ippsLPCToLSF_G723_16s (const Ipp16s* pSrcLPC, const Ipp16s*
   pSrcPrevLSF, Ipp16s* pDstLSF);
```

LSFToLPC_G723

Converts LSF coefficients to the LP coefficients.

```
IppStatus ippsLSFToLPC_G723_16s(const Ipp16s* pSrcLSF, Ipp16s* pDstLPC);
IppStatus ippsLSFToLPC_G723_16s_I (Ipp16s* pSrcLSFDstLPC);
```

LSFDecode_G723

Performs inverse quantization of LSFs.

```
IppStatus ippsLSFDecode_G723_16s (const Ipp16s* quantIndex, const
   Ipp16s* pPrevLSF, int erase, Ipp16s* pQuantLSF);
```

LSFQuant_G723

Quantizes LSF coefficients.

```
IppStatus ippsLSFQuant_G723_16s32s(const Ipp16s* pSrcLSF, const Ipp16s*
   pSrcPrevLSF, Ipp32s* pResultQLSFIndex);
```

OpenLoopPitchSearch_G723

Searches for an optimal pitch value.

```
IppStatus ippsOpenLoopPitchSearch_G723_16s(const Ipp16s*  pSrcWgtSpch,
   Ipp16s*  pResultOpenDelay);
```

ACELPFixedCodebookSearch_G723

Searches the ACELP fixed codebook for the excitation.

```
IppStatus ippsACELPFixedCodebookSearch_G723_16s(const Ipp16s*
   pSrcFixedCorr, const Ipp16s* pSrcMatrix, Ipp16s* pDstFixedSign,
   Ipp16s* pDstFixedPosition, Ipp16s* pResultGrid, Ipp16s*
   pDstFixedVector, Ipp16s* pSearchTimes);
IppStatus ippsACELPFixedCodebookSearch_G723_32s16s(const Ipp16s*
   pSrcFixedCorr, Ipp32s* pSrcDstMatrix, Ipp16s* pDstFixedSign, Ipp16s*
   pDstFixedPosition, Ipp16s* pResultGrid, Ipp16s* pDstFixedVector,
   Ipp16s* pSearchTimes);
```

AdaptiveCodebookSearch_G723

Searches for the close loop pitch and the adaptive gain index.

```
IppStatus ippsAdaptiveCodebookSearch_G723(Ipp16s valBaseDelay,
   const Ipp16s*  pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse,
   Ipp16s*  pSrcPrevExcitation, const Ipp32s* pSrcPrevError,
   Ipp16s*  pResultCloseLag, Ipp16s*  pResultAdptGainIndex, Ipp16s
   subFrame, Ipp16s sineDtct, IppSpchBitRate bitRate);
```

MPMLQFixedCodebookSearch_G723

Searches the MP-MLQ fixed codebook for the excitation.

```
ppStatus ippsMPMLQFixedCodebookSearch_G723(Ipp16s valBaseDelay, const
   Ipp16s* pSrcImpulseResponse, const Ipp16s* pSrcResidualTarget,
   Ipp16s* pDstFixedVector, Ipp16s* pResultGrid, Ipp16s*
   pResultTrainDirac, Ipp16s* pResultAmpIndex, Ipp16s* pResultAmplitude,
   Ipp32s* pResultPosition, Ipp16s subFrame);
```

ToeplizMatrix_G723

Calculates 416 elements of the Toepliz matrix for fixed codebook search.

```
IppStatus ippsToeplizMatrix_G723_16s(const Ipp16s*  pSrcImpulseResponse,
   Ipp16s*  pDstMatrix);
IppStatus ippsToeplizMatrix_G723_16s32s(const Ipp16s*
   pSrcImpulseResponse, Ipp32s*  pDstMatrix);
```

GainQuant_G723

Implements MP-MLQ gain estimation and quantization.

```
IppStatus ippsGainQuant_G723_16s (const Ipp16s* pImp, const Ipp16s* pSrc,
   Ipp16s* pDstLoc, Ipp16s* pDstAmp, Ipp32s* pMaxErr, Ipp16s* pGrid,
   Ipp16s* pAmp, int Np, int* isBest);
```

GainControl_G723

Extracts delayed pitch contribution.

```
IppStatus ippsGainControl_G723_16s_I (Ipp32s energy, Ipp16s* pSrcDst,
   Ipp16s* pGain);
```

HighPassFilter_G723

Performs high-pass filtering of the input signal.

```
IppStatus ippsHighPassFilter_G723_16s (const Ipp16s* pSrc, Ipp16s* pDst,
   int* pMem);
```

IIR16s_G723

Performs IIR filtering.

```
IppStatus ippsIIR16s_G723_16s32s (const Ipp16s* pCoeffs, const Ipp16s*
   pSrc, Ipp32s* pDst, Ipp16s* pMem);
IppStatus ippsIIR16s_G723_16s_I (const Ipp16s* pCoeffs, Ipp16s* pSrcDst,
   Ipp16s* pMem);
IppStatus ippsIIR16s_G723_32s16s_Sfs (const Ipp16s* pCoeffs, const
   Ipp32s* pSrc, int scaleFactor, Ipp16s* pDst, Ipp16s* pMem);
```

SynthesisFilter_G723
Computes the speech signal by filtering the input speech through the synthesis filter 1/A(z).

```
IppStatus ippsSynthesisFilter_G723_16s32s (const Ipp16s* pLPC, const
   Ipp16s* pSrc, Ipp32s* pDst, Ipp16s* pMem);
IppStatus ippsSynthesisFilter_G723_16s (const Ipp16s* pLPC, const
   Ipp16s*  pSrc, Ipp16s*  pMem, Ipp16s* pDst);
```

TiltCompensation_G723
Computes tilt compensation filter.

```
IppStatus ippsTiltCompensation_G723_32s16s (Ipp16s val, const Ipp32s*
   pSrc, Ipp16s* pDst);
```

HarmonicSearch_G723
Searches for the harmonic delay and gain for the harmonic noise shaping filter.

```
IppStatus ippsHarmonicSearch_G723_16s(Ipp16s valOpenDelay, const Ipp16s*
   pSrcWgtSpch, Ipp16s* pResultHarmonicDelay, Ipp16s*
   pResultHarmonicGain);
```

HarmonicNoiseSubtract_G723
Performs harmonic noise shaping.

```
IppStatus ippsHarmonicNoiseSubtract_G723_16s_I(Ipp16s val, int T, const
   Ipp16s* pSrc, Ipp16s* pSrcDst);
```

DecodeAdaptiveVector_G723
Restores the adaptive codebook vector from excitation, pitch, and adaptive gain.

```
IppStatus ippsDecodeAdaptiveVector_G723_16s(Ipp16s valBaseDelay, Ipp16s
   valCloseLag, Ipp16s valAdptGainIndex, const Ipp16s*
   pSrcPrevExcitation, Ipp16s* pDstAdptVector, IppSpchBitRate bitRate);
```

PitchPostFilter_G723
Calculates coefficients of the pitch post filter.

```
IppStatus ippsPitchPostFilter_G723_16s(Ipp16s valBaseDelay, const
   Ipp16s* pSrcResidual, Ipp16s* pResultDelay, Ipp16s* pResultPitchGain,
   Ipp16s* pResultScalingGain, Ipp16s subFrame, IppSpchBitRate bitRate);
```

GSM-AMR Related Functions

Interpolate_GSMAMR
Computes the weighted sum of two vectors.

```
IppStatus ippsInterpolate_GSMAMR_16s (const Ipp16s* pSrc1, const Ipp16s*
   pSrc2, Ipp16s* pDst, int len);
```

FFTFwd_RToPerm_GSMAMR
Computes the forward fast Fourier transform (FFT) of a real signal.

```
IppStatus ippsFFTFwd_RToPerm_GSMAMR_16s_I (Ipp16s* pSrcDst);
```

AutoCorr_GSMAMR

Estimates the autocorrelation sequence for a block of samples.

```
IppStatus ippsAutoCorr_GSMAMR_16s32s(const Ipp16s*  pSrcSpch, Ipp32s*
   pDstAutoCorr, IppSpchBitRate mode);
```

LevinsonDurbin_GSMAMR

Calculates the LP coefficients using the Levinson-Durbin algorithm.

```
IppStatus ippsLevinsonDurbin_GSMAMR(const Ipp32s*  pSrcAutoCorr,
   Ipp16s*  pSrcDstLpc);

IppStatus ippsLevinsonDurbin_GSMAMR_32s16s (const Ipp32s*  pSrcAutoCorr,
   Ipp16s*  pSrcDstLpc);
```

LPCToLSP_GSMAMR

Converts LP coefficients to line spectrum pairs.

```
IppStatus ippsLPCToLSP_GSMAMR_16s(const Ipp16s*  pSrcLpc, const Ipp16s*
   pSrcPrevLsp, Ipp16s*  pDstLsp);
```

LSPToLPC_GSMAMR

Converts LSPs to LP coefficients.

```
IppStatus ippsLSPToLPC_GSMAMR_16s(const Ipp16s*  pSrcLsp, Ipp16s*
   pDstLpc);
```

LSFToLSP_GSMAMR

Converts Line Spectral Frequencies to LSP coefficients.

```
IppStatus ippsLSFToLSP_GSMAMR_16s (const Ipp16s* pLSF, Ipp16s* pLSP) ;
```

LSPQuant_GSMAMR

Quantizes the LSP coefficient vector.

```
IppStatus ippsLSPQuant_GSMAMR_16s(const Ipp16s*  pSrcLsp, Ipp16s*
   pSrcDstPrevQLsfResidual, Ipp16s*  pDstQLsp, Ipp16s*  pDstQLspIndex,
   IppSpchBitRate  mode);
```

QuantLSPDecode_GSMAMR

Decodes quantized LSPs.

```
IppStatus ippsQuantLSPDecode_GSMAMR_16s(const Ipp16s*  pSrcQLspIndex,
   Ipp16s*  pSrcDstPrevQLsfResidual, Ipp16s*  pSrcDstPrevQLsf, Ipp16s*
   pSrcDstPrevQLsp, Ipp16s*  pDstQLsp, Ipp16s bfi, IppSpchBitRate mode);
```

OpenLoopPitchSearchNonDTX_GSMAMR

Computes the open-loop pitch lag.

```
IppStatus ippsOpenLoopPitchSearchNonDTX_GSMAMR_16s(const Ipp16s*
   pSrcWgtLpc1, const Ipp16s*  pSrcWgtLpc2, const Ipp16s*  pSrcSpch,
   Ipp16s*  pValResultPrevMidPitchLag, Ipp16s*  pValResultVvalue,
   Ipp16s*  pSrcDstPrevPitchLag, Ipp16s*  pSrcDstPrevWgtSpch,
   Ipp16s*  pDstOpenLoopLag, Ipp16s*  pDstOpenLoopGain, IppSpchBitRate
   mode);
```

OpenLoopPitchSearchDTXVAD1_GSMAMR

Extracts an open-loop pitch lag estimate (VAD 1 scheme is enabled).

```
IppStatus ippsOpenLoopPitchSearchDTXVAD1_GSMAMR_16s(const Ipp16s*
   pSrcWgtLpc1, const Ipp16s*  pSrcWgtLpc2, const Ipp16s*  pSrcSpch,
   Ipp16s*  pValResultToneFlag, Ipp16s*  pValResultPrevMidPitchLag,
   Ipp16s*  pValResultVvalue, Ipp16s*  pSrcDstPrevPitchLag,
   Ipp16s*  pSrcDstPrevWgtSpch, Ipp16s*  pResultMaxHpCorr, Ipp16s*
   pDstOpenLoopLag, Ipp16s*  pDstOpenLoopGain, IppSpchBitRate mode);
```

OpenLoopPitchSearchDTXVAD2_GSMAMR

Extracts an open-loop pitch lag estimate (VAD 2 scheme is enabled).

```
IppStatus ippsOpenLoopPitchSearchDTXVAD2_GSMAMR(const Ipp16s*
   pSrcWgtLpc1, const Ipp16s*  pSrcWgtLpc2, const Ipp16s*  pSrcSpch,
   Ipp16s*  pValResultPrevMidPitchLag, Ipp16s*  pValResultVvalue,
   Ipp16s*  pSrcDstPrevPitchLag, Ipp16s*  pSrcDstPrevWgtSpch,
   Ipp32s*  pResultMaxCorr, Ipp32s pResultWgtEnergy, Ipp16s*
   pDstOpenLoopLag, Ipp16s*  pDstOpenLoopGain, IppSpchBitRate mode);

IppStatus ippsOpenLoopPitchSearchDTXVAD2_GSMAMR_16s32s(const Ipp16s*
   pSrcWgtLpc1, const Ipp16s*  pSrcWgtLpc2, const Ipp16s*  pSrcSpch,
   Ipp16s*  pValResultPrevMidPitchLag, Ipp16s*  pValResultVvalue,
   Ipp16s*  pSrcDstPrevPitchLag, Ipp16s*  pSrcDstPrevWgtSpch, Ipp32s*
   pResultMaxCorr, Ipp32s pResultWgtEnergy, Ipp16s*  pDstOpenLoopLag,
   Ipp16s*  pDstOpenLoopGain, IppSpchBitRate mode);
```

ImpulseResponseTarget_GSMAMR

Computes the impulse response and target signal required for the adaptive codebook search.

```
IppStatus ippsImpulseResponseTarget_GSMAMR_16s(const Ipp16s*  pSrcSpch,
   const Ipp16s*  pSrcWgtLpc1, const Ipp16s*  pSrcWgtLpc2, const Ipp16s*
   pSrcQLpc, const Ipp16s*  pSrcSynFltState, const Ipp16s*
   pSrcWgtFltState, Ipp16s*  pDstImpulseResponse, Ipp16s*
   pDstLpResidual, Ipp16s*  pDstAdptTarget);
```

AdaptiveCodebookSearch_GSMAMR

Performs the adaptive codebook search.

```
IppStatus ippsAdaptiveCodebookSearch_GSMAMR_16s(const Ipp16s*
   pSrcTarget, const Ipp16s* pSrcImpulseResponse, Ipp16s*
   pSrcOpenLoopLag, Ipp16s*  pValResultPrevIntPitchLag, Ipp16s*
   pSrcDstExcitation, Ipp16s*  pResultFracPitchLag, Ipp16s*
   pResultAdptIndex, Ipp16s*  pDstAdptVector, Ipp16s subFrame,
   IppSpchBitRate mode);
```

AdaptiveCodebookDecode_GSMAMR

Decodes the adaptive codebook parameters.

```
IppStatus ippsAdaptiveCodebookDecode_GSMAMR_16s(Ipp16s valAdptIndex,
   Ipp16s*  pValResultPrevIntPitchLag, Ipp16s*  pValResultLtpLag,
   Ipp16s*  pSrcDstExcitation, Ipp16s*  pResultIntPitchLag,
   Ipp16s*  pDstAdptVector, Ipp16s subFrame, Ipp16s bfi,
   Ipp16s inBackgroundNoise, Ipp16s voicedHangover, IppSpchBitRate
   mode);
```

AdaptiveCodebookGain_GSMAMR

Calculates the gain of the adaptive-codebook vector and the filtered codebook vector.

```
IppStatus ippsAdaptiveCodebookGain_GSMAMR_16s (const Ipp16s*
   pSrcAdptTarget, const Ipp16s* pSrcFltAdptVector, Ipp16s*
   pResultAdptGain);

IppStatus ippsAdaptiveCodebookGainCoeffs_GSMAMR_16s (const Ipp16s*
   pSrcAdptTarget, const Ipp16s* pSrcFltAdptVector, Ipp16s*
   pResultAdptGain, Ipp16s* pResultAdptGainCoeffs);
```

AlgebraicCodebookSearch_GSMAMR

Searches the algebraic codebook.

```
IppStatus ippsAlgebraicCodebookSearch_GSMAMR_16s(Ipp16s valIntPitchLag,
   Ipp16s valBoundQAdptGain, const Ipp16s*  pSrcFixedTarget,
   const Ipp16s*  pSrcLtpResidual, Ipp16s*  pSrcDstImpulseResponse,
   Ipp16s*  pDstFixedVector, Ipp16s*  pDstFltFixedVector,
   Ipp16s*  pDstEncPosSign, Ipp16s subFrame, IppSpchBitRate mode);

IppStatus ippsAlgebraicCodebookSearchEX_GSMAMR_16s(Ipp16s
   valIntPitchLag, Ipp16s valBoundQAdptGain, const Ipp16s*
   pSrcFixedTarget, const Ipp16s*  pSrcLtpResidual, Ipp16s*
   pSrcDstImpulseResponse, Ipp16s* pDstFixedVector, Ipp16s*
   pDstFltFixedVector, Ipp16s*  pDstEncPosSign, Ipp16s subFrame,
   IppSpchBitRate mode, Ipp32s*  pBuffer);
```

FixedCodebookDecode_GSMAMR

Decodes the fixed codebook vector.

```
IppStatus ippsFixedCodebookDecode_GSMAMR_16s(const Ipp16s*
   pSrcFixedIndex, Ipp16s*  pDstFixedVector, Ipp16s subFrame,
   IppSpchBitRate mode);
```

Preemphasize_GSMAMR

Computes pre-emphasis of an input signal in VAD option 2.

```
IppStatus ippsPreemphasize_GSMAMR_16s (Ipp16s gamma, const Ipp16s* pSrc,
   Ipp16s* pDst, int len, Ipp16s* pMem);
```

VAD1_GSMAMR

Implements the VAD functionality corresponding to VAD option 1.

```
IppStatus ippsVAD1_GSMAMR_16s(const Ipp16s pSrcSpch, IppGSMAMRVad1State*
   pValResultVad1State, Ipp16s*  pResultVadFlag, Ipp16s maxHpCorr,
   Ipp16s toneFlag);
```

VAD2_GSMAMR

Implements the VAD functionality corresponding to VAD option 2.

```
IppStatus ippsVAD2_GSMAMR_16s(const Ipp16s*  pSrcSpch,
   IppGSMAMRVad2State*  pValResultVad2State, Ipp16s*  pResultVadFlag,
   Ipp16s ltpFlag);
```

EncDTXSID_GSMAMR
Extracts parameters for the SID frame.

```
IppStatus ippsEncDTXSID_GSMAMR_16s(const Ipp16s*  pSrcLspBuffer, const
    Ipp16s*  pSrcLogEnergyBuffer, Ipp16s*  pValResultLogEnergyIndex,
    Ipp16s*  pValResultDtxLsfRefIndex, Ipp16s*  pSrcDstQLsfIndex,
    Ipp16s*  pSrcDstPredQErr, Ipp16s*  pSrcDstPredQErrMR122, Ipp16s
    sidFlag);
```

EncDTXHandler_GSMAMR
Determines the SID flag of current frame.

```
IppStatus ippsEncDTXHandler_GSMAMR_16s(Ipp16s*  pValResultHangOverCount,
    Ipp16s* pValResultDtxElapsedCount, Ipp16s*  pValResultUsedMode,
    Ipp16s*  pResultSidFlag, Ipp16s vadFlag);
```

EncDTXBuffer_GSMAMR , DecDTXBuffer_GSMAMR
Buffer the LSP (or LSF) coefficients and previous log energy coefficients.

```
IppStatus ippsEncDTXBuffer_GSMAMR_16s(const Ipp16s*  pSrcSpch,
    const Ipp16s*  pSrcLsp, Ipp16s* pValResultUpdateIndex,
    Ipp16s*  pSrcDstLspBuffer, Ipp16s*  pSrcDstLogEnergyBuffer);

IppStatus  ippsDecDTXBuffer_GSMAMR_16s(const Ipp16s*  pSrcSpch,
    const Ipp16s*  pSrcLsf, Ipp16s* pValResultUpdateIndex,
    Ipp16s* pSrcDstLsfBuffer, Ipp16s*  pSrcDstLogEnergyBuffer);
```

PostFilter_GSMAMR
Filters the synthesized speech.

```
IppStatus ippsPostFilter_GSMAMR_16s(const Ipp16s*  pSrcQLpc,
    const Ipp16s*  pSrcSpch, Ipp16s*  pValResultPrevResidual,
    Ipp16s*  pValResultPrevScalingGain, Ipp16s*  pSrcDstFormantFIRState,
    Ipp16s*  pSrcDstFormantIIRState, Ipp16s*  pDstFltSpch, IppSpchBitRate
    mode);
```

AMR Wideband Related Functions

ResidualFilter_AMRWB
Computes the LPC residual

```
IppStatus ippsResidualFilter_AMRWB_16s_Sfs (const Ipp16s* pSrcLpc, int
    order, const Ipp16s* pSrcSpeech, Ipp16s* pDstResidualSignal, int len,
    int scaleFactor);
```

LPCToISP_AMRWB
Performs LP to ISP coefficients conversion.

```
IppStatus ippsLPCToISP_AMRWB_16s( const Ipp16s* pSrcLpc, Ipp16s*
    pDstIsp,  const Ipp16s* pSrcPrevIsp);
```

ISPToLPC_AMRWB
Performs ISP to LP coefficients conversion.

```
IppStatus ippsISPToLPC_AMRWB_16s(const Ipp16s* pSrcIsp, Ipp16s* pDstLpc,
   int len);
```

ISPToISF_Norm_AMRWB
Performs ISP to ISF coefficients conversion.

```
IppStatus ippsISPToISF_Norm_AMRWB_16s(const Ipp16s* pSrcIsp, Ipp16s*
   pDstIsf, int len);
```

ISFToISP_AMRWB
Performs ISF conversion to ISP.

```
IppStatus ippsISFToISP_AMRWB_16s(const Ipp16s* pSrcIsf, Ipp16s* pDstIsp,
   int  len);
```

OpenLoopPitchSearch_AMRWB
Extracts an open-loop pitch lag estimate from the weighted input speech.

```
IppStatus ippsOpenLoopPitchSearch_AMRWB_16s(const Ipp16s* pSrcWgtSpch,
   const Ipp16s* pSrcFltWgtSpch, Ipp16s* pPrevMidPitchLag, Ipp16s*
   pAdaptiveParam, Ipp16s* pDstOpenLoopLag, Ipp16s* pToneFlag, Ipp16s*
   pDstOpenLoopGain, Ipp16s* pSrcDstPrevPitchLag, Ipp16s*
   pSrcDstLagSwitcher, int len );
```

HighPassFilterGetSize_AMRWB
Returns the size of the high-pass filter state memory.

```
IppStatus ippsHighPassFilterGetSize_AMRWB_16s (int order, int*
   pDstSize);
```

HighPassFilterInit_AMRWB
Initializes the state memory of high-pass filter.

```
IppStatus ippsHighPassFilterInit_AMRWB_16s(Ipp16s* pFilterCoeffA,
   Ipp16s* pFilterCoeffB, int order, IppsHighPassFilterState_AMRWB_16s*
   pState);
```

HighPassFilter_AMRWB
Performs high-pass filtering.

```
IppStatus ippsHighPassFilter_AMRWB_16s_Sfs (const Ipp16s* pSrc, Ipp16s*
   pDst, int len, IppsHighPassFilterState_AMRWB_16s* pState, int
   scaleFactor);

IppStatus ippsHighPassFilter_AMRWB_16s_ISfs (Ipp16s* pSrcDst, int len,
   IppsHighPassFilterState_AMRWB_16s* pState, int scaleFactor);

IppStatus ippsHighPassFilter_Direct_AMRWB_16s (const Ipp16s* pSrcCoeff,
   const Ipp16s* pSrc, Ipp16s* pDst, int len, int borderMode);
```

Preemphasize_AMRWB
Computes pre-emphasis of a speech signal.

```
IppStatus ippsPreemphasize_AMRWB_16s_ISfs (Ipp16s gamma, Ipp16s*
   pSrcDst, int len, int scaleFactor,  Ipp16s* pMem);
```

Deemphasize_AMRWB

Performs de-emphasis filtering.

```
IppStatus ippsDeemphasize_AMRWB_NR_16s_I(Ipp16s gamma, Ipp16s* pSrcDst,
   Ipp16s len, Ipp16s* pMem);
IppStatus ippsDeemphasize_AMRWB_32s16s(Ipp16s gamma, const Ipp32s* pSrc,
   Ipp16s* pDst, int len, Ipp16s* pMem);
```

SynthesisFilter_AMRWB

Reconstructs the speech signal from LP coefficients and residuals.

```
IppStatus ippsSynthesisFilter_AMRWB_16s32s_I(const Ipp16s* pSrcLpc, int
   order,  const Ipp16s* pSrcExc, Ipp32s* pSrcDstSignal, int len);
```

VADGetSize_AMRWB

Returns the size of the VAD module state memory.

```
IppStatus ippsVADGetSize_AMRWB_16s (int* pDstSize);
```

VADInit_AMRWB

Initializes the VAD module state memory.

```
IppStatus ippsVADInit_AMRWB_16s(IppsVADState_AMRWB_16s* pState);
```

VAD_AMRWB

Performs VAD in AMR WB encoder.

```
IppStatus ippsVAD_AMRWB_16s(const Ipp16s* pSrcSpch,
   IppsVADState_AMRWB_16s* pSrcDstVadState, Ipp16s* pToneFlag, Ipp16s*
   pVadFlag);
```

AlgebraicCodebookSearch_AMRWB

Performs the fixed (algebraic) codebook search.

```
IppStatus ippsAlgebraicCodebookSearch_AMRWB_16s(const Ipp16s*
   pSrcFixedTarget, const Ipp16s* pSrcLtpResidual, Ipp16s*
   pSrcDstImpulseResponse, Ipp16s* pDstFixedVector, Ipp16s*
   pDstFltFixedVector, IppSpchBitRate mode, Ipp16s* pDstIndex);
```

AlgebraicCodebookDecode_AMRWB

Decodes the fixed (algebraic) codebook indexes.

```
IppStatus ippsAlgebraicCodebookDecode_AMRWB_16s (const Ipp16s* pSrcIdxs,
   Ipp16s* pDstFixedCode, IppSpchBitRate mode);
```

AdaptiveCodebookGainCoeff_AMRWB

Computes the adaptive codebook gain.

```
IppStatus ippsAdaptiveCodebookGainCoeff_AMRWB_16s(const Ipp16s*
   pSrcAdptTarget, const Ipp16s*  pSrcFltAdptVector, Ipp16s*
   pAdptGainCoeffs, Ipp16s*  pResultAdptGain, int len);
```

AdaptiveCodebookSearch_AMRWB

Performs the adaptive codebook search.

```
IppStatus ippsAdaptiveCodebookSearch_AMRWB_16s(const Ipp16s*
   pSrcAdptTarget, const Ipp16s*  pSrcImpulseResponse, const Ipp16s*
   pSrcOpenLoopLag, Ipp16s*  pPitchLag, Ipp16s*  pPitchLagBounds,
   Ipp16s*  pSrcDstExcitation, Ipp16s*  pFracPitchLag, Ipp16s*
   pAdptIndex, int   subFrame, IppSpchBitRate mode);
```

AdaptiveCodebookDecodeGetSize_AMRWB

Queries the memory length of the adaptive codebook decode module.

```
IppStatus ippsAdaptiveCodebookDecodeGetSize_AMRWB_16s(int* pDstSize);
```

AdaptiveCodebookDecodeInit_AMRWB

Initializes the adaptive codebook decode module memory.

```
IppStatus ippsAdaptiveCodebookDecodeInit_AMRWB_16s(
   IppsAdaptiveCodebookDecodeState_AMRWB_16s* pState);
```

AdaptiveCodebookDecodeUpdate_AMRWB

Updates the adaptive codebook decode module memory.

```
IppStatus ippsAdaptiveCodebookDecodeUpdate_AMRWB_16s(Ipp16s
   valIntPitchLag, Ipp16s valPitchGain,
   IppsAdaptiveCodebookDecodeState_AMRWB_16s* pState);
```

AdaptiveCodebookDecode_AMRWB

Performs the adaptive codebook decoding.

```
IppStatus ippsAdaptiveCodebookDecode_AMRWB_16s(int valAdptIndex, Ipp16s*
   pResultFracPitchLag, Ipp16s* pSrcDstExcitation, Ipp16s* pPitchLag,
   Ipp16s* pPitchLagBounds, int subFrame, int bfi, int unusableFrame,
   IppSpchBitRate mode, IppsAdaptiveCodebookDecodeState_AMRWB_16s*
   pState);
```

ISFQuant_AMRWB

Quantizes the ISF.

```
IppStatus ippsISFQuant_AMRWB_16s(const Ipp16s* pSrcIsf, Ipp16s*
   pSrcDstResidual, Ipp16s* pDstQIsf, Ipp16s* pDstQIsfIndex,
   IppSpchBitRate mode);
```

ISFQuantDecode_AMRWB

Decodes quantized ISFs from the received codebook index.

```
IppStatus ippsISFQuantDecode_AMRWB_16s(const Ipp16s* pSrcIdxs, Ipp16s*
   pDstQntIsf, Ipp16s* pSrcDstResidual, const Ipp16s* pSrcPrevQntIsf,
   Ipp16s* pSrcDstIsfMemory, int bfi, IppSpchBitRate mode);
```

ISFQuantDTX_AMRWB

Quantizes the ISF coefficient vector in case of DTX mode.

```
IppStatus ippsISFQuantDTX_AMRWB_16s(const Ipp16s* pSrcIsf, Ipp16s*
   pDstQntIsf, Ipp16s* pDstIdxs);
```

ISFQuantDecodeDTX_AMRWB
Decodes quantized ISFs in case of DTX mode.

```
IppStatus ippsISFQuantDecodeDTX_AMRWB_16s(const Ipp16s* pSrcIdxs,
   Ipp16s* pDstQntIsf);
```

GainQuant_AMRWB
Quantizes the adaptive codebook gains.

```
IppStatus ippsGainQuant_AMRWB_16s(const Ipp16s* pSrcAdptTarget, const
   Ipp16s* pSrcFltAdptVector, int valFormat, const Ipp16s*
   pSrcFixedVector, const Ipp16s* pSrcFltFixedVector, const Ipp16s*
   pSrcCorr, Ipp16s* pSrcDstEnergyErr, Ipp16s* pSrcDstPitchGain, int*
   pDstCodeGain, int valClipFlag, Ipp16s* pDstQGainIndex, int lenSrc,
   IppSpchBitRate mode);
```

DecodeGain_AMRWB
Decodes adaptive and fixed-codebook gains.

```
IppStatus ippsDecodeGain_AMRWB_16s(int valQIndex, Ipp32s valEnergy,
   Ipp16s* pDstPitchGain, int* pDstCodeGain, int bfi, int prevBfi,
   Ipp16s* pSrcDstPastEnergy, Ipp16s*  pPrevCodeGain, Ipp16s*
   pSrcDstPastCodeGain, IppSpchBitRate mode);
```

EncDTXBuffer_AMRWB
Buffers the ISP coefficients and previous logarithm energy coefficients.

```
IppStatus ippsEncDTXBuffer_AMRWB_16s(const Ipp16s* pSrcSpch, const
   Ipp16s* pSrcIsp, Ipp16s* pUpdateIndex, Ipp16s* pSrcDstIspBuffer,
   Ipp16s* pSrcDstLogEnergyBuffer, IppSpchBitRate mode);
```

DecDTXBuffer_AMRWB
Buffers the ISF coefficients and previous logarithm energy coefficients.

```
IppStatus ippsDecDTXBuffer_AMRWB_16s(const Ipp16s* pSrcSpch, const
   Ipp16s* pSrcIsf, Ipp16s* pUpdateIndex, Ipp16s* pSrcDstIsfBuffer,
   Ipp16s* pSrcDstLogEnergyBuffer);
```

GSM Full Rate Related Functions

RPEQuantDecode_GSMFR
Performs APCM inverse quantization.

```
IppStatus ippsRPEQuantDecode_GSMFR_16s (const Ipp16s* pSrc, Ipp16s ampl,
   Ipp16s amplSfs, Ipp16s* pDst);
```

Deemphasize_GSMFR
Performs de-emphasis filtering.

```
IppStatus ippsDeemphasize_GSMFR_16s_I (Ipp16s* pSrcDst, int len,
   Ipp16s* pMem);
```

ShortTermAnalysisFilter_GSMFR
Performs short-term analysis filtering.

```
IppStatus ippsShortTermAnalysisFilter_GSMFR_16s_I (const Ipp16s* pRC,
   Ipp16s* pSrcDstSpch, int len, Ipp16s* pMem);
```

ShortTermSynthesisFilter_GSMFR
Performs short-term synthesis filtering.

```
IppStatus ippsShortTermSynthesisFilter_GSMFR_16s (const Ipp16s* pRC,
   const Ipp16s* pSrcResidual, Ipp16s* pDstSpch, int len, Ipp16s* pMem);
```

HighPassFilter_GSMFR
Performs high-pass filtering of the input speech signal.

```
IppStatus ippsHighPassFilter_GSMFR_16s (const Ipp16s* pSrc, Ipp16s*
   pDst, int len, int* pMem);
```

Schur_GSMFR
Estimates the reflection coefficients by Schur recursion.

```
IppStatus ippsSchur_GSMFR_32s16s (const Ipp32s* pSrc, Ipp16s* pDst,
   int dstLen);
```

WeightingFilter_GSMFR
Calculates the weighting filter.

```
IppStatus ippsWeightingFilter_GSMFR_16s (const Ipp16s* pSrc, Ipp16s*
   pDst, int dstLen);
```

Preemphasize_GSMFR
Computes pre-emphasis of a speech signal.

```
IppStatus ippsPreemphasize_GSMFR_16s(const Ipp16s* pSrc, Ipp16s* pDst,
   int* pMem, int len);
```

G.722.1 Related Functions

DCTFwd_G722, DCTInv_G722
Computes the forward or inverse discrete cosine transform (DCT) of a signal.

```
IppStatus ippsDCTFwd_G722_16s (const Ipp16s* pSrc, Ipp16s* pDst);
IppStatus ippsDCTInv_G722_16s (const Ipp16s* pSrc, Ipp16s* pDst);
```

DecomposeMLTToDCT
Decomposes the MLT transform input signal to the form of the DCT input signal.

```
IppStatus ippsDecomposeMLTToDCT_G722_16s(const Ipp16s* pSrcSpch, Ipp16s*
   pSrcDstSpchOld, Ipp16s* pDstSpchDecomposed);
```

DecomposeDCTToMLT
Decomposes IDCT output signal to the form of the MLT transform output signal.

```
IppStatus ippsDecomposeDCTToMLT_G722_16s(const Ipp16s*
   pSrcSpchDecomposed, Ipp16s* pSrcDstSpchDecomposedOld, Ipp16s*
   pDstSpch);
```

HuffmanEncode_G722
Performs Huffman encoding of the quantized amplitude envelope indexes.

```
IppStatus ippsHuffmanEncode_G722_16s32u(int category, int
   qntAmpEnvIndex, const Ipp16s* pSrcMLTCoeffs, Ipp32u* pDstCode, int*
   pCodeLength);
```

G.726 Related Functions

EncodeGetStateSize_G726
Informative function, returns the number of bytes needed for encoder memory.

```
IppStatus ippsEncodeGetStateSize_G726_16s8u (unsigned int* pEncSize);
```

EncodeInit_G726
Initializes the memory for the ADPCM encoder.

```
IppStatus ippsEncodeInit_G726_16s8u (IppsEncoderState_G726_16s* pEncMem,
   IppSpchBitRate rate);
```

Encode_G726
Performs ADPCM compression of the uniform PCM input.

```
IppStatus ippsEncode_G726_16s8u (IppsEncoderState_G726_16s* pEncMem,
   const Ipp16s* pSrc, Ipp8u* pDst, unsigned int len);
```

DecodeGetStateSize_G726
Informative function, returns the number of bytes needed for decoder memory.

```
IppStatus ippsDecodeGetStateSize_G726_8u16s (unsigned int* pDecSize);
```

DecodeInit_G726
Initializes the memory for the G726 decoder.

```
IppStatus ippsDecodeInit_G726_8u16s (IppsDecoderState_G726_16s* pDecMem,
   IppSpchBitRate rate, IppPCMLaw law);
```

Decode_G726
Performs decompression of the ADPCM bit-stream.

```
IppStatus ippsDecode_G726_8u16s (IppsDecoderState_G726_16s* pDecMem,
   const Ipp8u* pSrc, Ipp16s* pDst, unsigned int len)
```

G.728 Related Functions

IIRGetStateSize_G728
Gets the size of IIR state structure to be used.

```
IppStatus ippsIIR16sGetStateSize_G728_16s (int* pSize);
```

IIR_Init_G728
Initializes the IIR state structure.

```
IppStatus ippsIIR16sInit_G728_16s (IppsIIRState16s_G728_16s* pMem );
```

IIR_G728

Applies IIR filter to multiple samples.

```
IppStatus ippsIIR16s_G728_16s (const Ipp16s* pCoeffs,
   const Ipp16s* pSrcQntSpeech, Ipp16s* pDstWgtSpeech, int len,
   IppsIIRState16s_G728_16s* pMem );
```

SynthesisFilterGetStateSize_G728

Gets the size of synthesis filter state structure.

```
IppStatus ippsSynthesisFilterGetStateSize_G728_16s (int* pSize);
```

SynthesisFilterInit_G728

Initializes the synthesis filter state structure.

```
IppStatus ippsSynthesisFilterInit_G728_16s
   (IppsSynthesisFilterState_G728_16s* pMem);
```

SyntesisFilter_G728

Applies the synthesis filter to multiple samples.

```
IppStatus ippsSyntesisFilterZeroInput_G728_16s (const Ipp16s* pCoeffs,
   Ipp16s* pSrcDstExc, Ipp16s excSfs, Ipp16s* pDstSpeech, Ipp16s*
   pSpeechSfs, IppsSynthesisFilterState_G728_16s* pMem);
```

CombinedFilterGetStateSize_G728

Gets the size of combined filter state structure.

```
IppStatus ippsCombinedFilterGetStateSize_G728_16s (int* pSize);
```

CombinedFilterInit_G728

Initializes the combined filter state structure.

```
IppStatus ippsCombinedFilterInit_G728_16s
   (IppsCombinedFilterState_G728_16s* pMem);
```

CombinedFilter_G728

Applies the combined filter to multiple samples.

```
IppStatus ippsCombinedFilterZeroInput_G728_16s(const Ipp16s* pSyntCoeff,
   const Ipp16s* pWgtCoeff, Ipp16s* pDstWgtZIR,
   IppsCombinedFilterState_G728_16s* pMem);

IppStatus ippsCombinedFilterZeroState_G728_16s(const Ipp16s* pSyntCoeff,
   const Ipp16s* pWgtCoeff, Ipp16s* pSrcDstExc, Ipp16s excSfs, Ipp16s*
   pDstSpeech, Ipp16s* pSpeechSfs, IppsCombinedFilterState_G728_16s*
   pMem);
```

PostFilterGetStateSize_G728

Gets the size of the post filter state structure.

```
IppStatus ippsPostFilterGetStateSize_G728_16s (int* pSize);
```

PostFilterInit_G728

Initializes the post filter state structure.

```
IppStatus ippsPostFilterInit_G728_16s (IppsPostFilterState_G728_16s*
   pMem);
```

PostFilter_G728
Applies the post filter to multiple samples.

```
IppStatus ippsPostFilter_G728_16s (Ipp16s gl, Ipp16s glb, Ipp16s kp, Ipp16s
   tiltz, const Ipp16s* pCoeffs, const Ipp16s* pSrc, Ipp16s* pDst,
   IppsPostFilterState_G728_16s* pMem);
```

PostFilterAdapterGetStateSize_G728
Gets the size of the IppsPostFilterAdapterState structure to be used.

```
IppStatus  ippsPostFilterAdapterGetStateSize_G728 (int* pSize);
```

PostFilterAdapterStateInit_G728
Initializes the IppsPostFilterAdapterState structure.

```
IppStatus  ippsPostFilterAdapterStateInit_G728 (
   IppsPostFilterAdapterState* pMem);
```

LPCInverseFilter_G728
Computes the LPC prediction residual.

```
IppStatus ippsLPCInverseFilter_G728_16s (const Ipp16s* pSrcSpeech, const
   Ipp16s* pCoeffs, Ipp16s* pDstResidual, IppsPostFilterAdapterState_G728*
   pMem);
```

PitchPeriodExtraction_G728
Extracts pitch period from the LPC prediction residual.

```
IppStatus ippsPitchPeriodExtraction_G728_16s (const Ipp16s* pSrcResidual,
   int* pPitchPeriod, IppsPostFilterAdapterState_G728* pMem);
```

WinHybridGetStateSize_G728
Gets the size of hybrid windowing module state structure.

```
IppStatus ippsWinHybridGetStateSize_G728_16s (int M, int L, int N, int DIM,
   int* pSize);
```

WinHybridInit_G728
Initializes the hybrid windowing module state structure.

```
IppStatus ippsWinHybridInit_G728_16s (const Ipp16s* pWinTab, int M, int L,
   int N, int DIM, Ipp16s a2L, IppsWinHybridState_G728_16s* pMem);
```

WinHybrid_G728
Applies the hybrid windowing.

```
IppStatus ippsWinHybridBlock_G728_16s(Ipp16s bfi, const Ipp16s* pSrc, const
   Ipp16s* pSrcSfs, Ipp16s* pDst, IppsWinHybridState_G728_16s* pMem);
IppStatus ippsWinHybrid_G728_16s(Ipp16s bfi, const Ipp16s* pSrc, const
   Ipp16s* pSrcSfs, Ipp16s* pDst, IppsWinHybridState_G728_16s* pMem);
```

LevinsonDurbin_G728
Calculates LP coefficients from the autocorrelation coefficients.

```
IppStatus ippsLevinsonDurbin_G728_16s_Sfs(const Ipp16s* pSrcAutoCorr, int
   order, Ipp16s* pDstLPC, Ipp16s* pDstResidualEnergy, Ipp16s*
   pDstScaleFactor);
```

```
IppStatus ippsLevinsonDurbin_G728_16s_ISfs(const Ipp16s* pSrcAutoCorr,
   int numSrcLPC, int order, Ipp16s* pSrcDstLPC, Ipp16s*
   pSrcDstResidualEnergy, Ipp16s* pSrcDstScaleFactor);
```

CodebookSearch_G728

Searches the codebook for the best code vector.

```
IppStatus ippsCodebookSearch_G728_16s(const Ipp16s* pSrcCorr, const
   Ipp16s* pSrcEnergy, int* pDstShapeIdx, int* pDstGainIdx, short*
   pDstCodebookIdx, IppSpchBitRate rate);
```

ImpulseResponseEnergy_G728

Implements shape codebook vector convolution and energy calculation.

```
IppStatus ippsImpulseResponseEnergy_G728_16s(const Ipp16s* pSrcImpResp,
   Ipp16s* pDstEnergy);
```

Echo Canceller Related Functions

SubbandProcessGetSize

Returns size of the subband process state structure.

```
IppStatus ippsSubbandProcessGetSize_32f(int order, int windowLen, int*
   pStateSize, int* pInitBufSize, int* pBufSize);
IppStatus ippsSubbandProcessGetSize_16s(int order, int windowLen, int*
   pStateSize, int* pInitBufSize, int* pBufSize);
```

SubbandProcessInit

Initializes the subband process state structure.

```
IppStatus ippsSubbandProcessInit_32f(IppsSubbandProcessState_32f*
   pState, int order, int frameSize, int windowLen, const Ipp32f*
   pWindow, Ipp8u* pInitBuf);
IppStatus ippsSubbandProcessInit_16s(IppsSubbandProcessState_16s*
   pState, int order, int frameSize, int windowLen, const Ipp32s*
   pWindow, Ipp8u* pInitBuf);
```

SubbandAnalysis

Decomposes a frame into a complex subband representation.

```
IppStatus ippsSubbandAnalysis_32f32fc(const Ipp32f* pSignal, Ipp32fc*
   pSubbands, IppsSubbandProcessState_32f* pState, Ipp8u* pBuf);
IppStatus ippsSubbandAnalysis_16s32sc_Sfs(const Ipp16s* pSignal,
   Ipp32sc* pSubbands, IppsSubbandProcessState_16s* pState, int
   scaleFactor, Ipp8u* pBuf);
```

SubbandSynthesis

Reconstructs frame from a complex subband representation.

```
IppStatus ippsSubbandSynthesis_32fc32f(const Ipp32fc* pSubbands, Ipp32f*
   pSignal, IppsSubbandProcessState_32f* pState, Ipp8u* pBuf);
```

```
IppStatus ippsSubbandSynthesis_32sc16s_Sfs(const Ipp32sc* pSubbands,
    Ipp16s* pSignal, IppsSubbandProcessState_16s* pState, int scaleFactor,
    Ipp8u* pBuf);
```

SubbandControllerGetSize_EC

Returns size of the subband controller state structure.

```
IppStatus ippsSubbandControllerGetSize_EC_32f(int numSubbands, int
    frameSize, int numSegments, ippECFrequency sampleFreq, int* pSize);
IppStatus ippsSubbandControllerGetSize_EC_16s(int numSubbands, int
    frameSize, int numSegments, ippECFrequency sampleFreq, int* pSize);
```

SubbandControllerInit_EC

Initializes the subband controller state structure.

```
IppStatus
    ippsSubbandControllerInit_EC_32f(IppsSubbandControllerState_EC_32f*
    pState, int numSubbands, int frameSize, int numSegments, ippECFrequency
    sampleFreq);
IppStatus
    ippsSubbandControllerInit_EC_16s(IppsSubbandControllerState_EC_16s*
    pState, int numSubbands, int frameSize, int numSegments, ippECFrequency
    sampleFreq);
```

SubbandControllerUpdate_EC

Updates controller state and returns the step sizes.

```
IppStatus ippsSubbandControllerUpdate_EC_32f(const Ipp32f* pSrcRin, const
    Ipp32f* pSrcSin, const Ipp32fc** ppSrcRinSubbandsHistory, const Ipp32fc*
    pSrcSinSubbands, double* pDstStepSize,
    IppsSubbandControllerState_EC_32f* pState);
IppStatus ippsSubbandControllerUpdate_EC_16s(const Ipp16s* pSrcRin, const
    Ipp16s* pSrcSin, const Ipp32sc** ppSrcRinSubbandsHistory, const Ipp32sc*
    pSrcSinSubbands, IppAECScaled32s* pDstStepSize,
    IppsSubbandControllerState_EC_16s* pState);
```

SubbandController_EC

The main subband controller function which updates filter coefficients and returns output gain coefficients.

```
IppStatus ippsSubbandController_EC_32f(const Ipp32fc*
    pSrcAdaptiveFilterErr, const Ipp32fc* pSrcFixedFilterErr, Ipp32fc**
    ppDstAdaptiveCoefs, Ipp32fc** ppDstFixedCoefs, Ipp32f* pDstSGain,
    IppsSubbandControllerState_EC_32f* pState);
IppStatus ippsSubbandController_EC_16s(const Ipp32sc*
    pSrcAdaptiveFilterErr, const Ipp32sc* pSrcFixedFilterErr, Ipp32sc**
    pDstpAdaptiveCoefs, Ipp32sc** ppFixedCoefs, Ipp32s* pDstSGain,
    IppsSubbandControllerState_EC_16s* pState);
```

SubbandControllerReset_EC

Resets the subband controller state.

```
IppStatus
   ippsSubbandControllerReset_EC_32f(IppsSubbandControllerState_EC_32f*
   pState);
IppStatus
   ippsSubbandControllerReset_EC_16s(IppsSubbandControllerState_EC_16s*
   pState);
```

ToneDetectGetStateSize_EC

Returns size of the tone detector state structure.

```
IppStatus ippsToneDetectGetStateSize_EC_16s(ippECFrequency sampleFreq,
   int* pSize);
IppStatus ippsToneDetectGetStateSize_EC_32f(ippECFrequency sampleFreq,
   int* pSize);
```

ToneDetectInit_EC

Initializes the tone detector state structure.

```
IppStatus ippsToneDetectInit_EC_16s(IppsToneDetectState_EC_16s* pState,
   ippECFrequency sampleFreq);
IppStitus ippsToneDetectInit_EC_32f(IppsToneDetectState_EC_32f* pState,
   ippECFrequency sampleFreq);
```

ToneDetect_EC

Detects the signal of 2100 Hz frequency with every 450 ms phase reversal.

```
IppStatus ippsToneDetect_EC_16s(const Ipp16s* pSignal, int len, int*
   pResult, IppsToneDetectState_EC_16s* pState);
IppStatus ippsToneDetect_EC_32f(const Ipp32f* pSignal, int len, int*
   pResult, IppsToneDetectState_EC_32f* pState);
```

FullbandControllerGetSize_EC

Returns size of the fullband controller state structure.

```
IppStatus ippsFullbandControllerGetSize_EC_32f(int frameSize, int tapLen,
   ippECFrequency sampleFreq, int* pSize);
IppStatus ippsFullbandControllerGetSize_EC_16s(int frameSize, Int tapLen,
   ippECFrequency sampleFreq, int* pSize);
```

FullbandControllerInit_EC

Initializes the fullband controller state structure.

```
IppStatus
   ippsFullbandControllerInit_EC_32f(IppsFullbandControllerState_EC_32f*
   pState, int frameSize, int tapLen, ippECFrequency sampleFreq);
IppStatus
   ippsFullbandControllerInit_EC_16s(IppsFullbandControllerState_EC_16s*
   pState, int frameSize, int tapLen, ippECFrequency sampleFreq);
```

FullbandControllerUpdate_EC

Updates the fullband controller state and returns the step sizes.

```
IppStatus ippsFullbandControllerUpdate_EC_32f(const Ipp32f* pSrcRin,
   const Ipp32f* pSrcSin, Ipp32f* pDstStepSize,
   IppsFullbandControllerState_EC_32f* pState);

IppStatus ippsFullbandControllerUpdate_EC_16s(const Ipp16s* pSrcRin,
   const Ipp16s* pSrcSin, IppAECScaled32s* pDstStepSize,
   IppsFullbandControllerState_EC_16s* pState);
```

FullbandController_EC

The main fullband controller function which updates filter coefficients and returns output gain coefficients.

```
IppStatus ippsFullbandController_EC_32f(const Ipp32f* pAdaptiveFilterErr,
   const Ipp32f* pFixedFilterErr, Ipp32f* pAdaptiveCoefs, Ipp32f*
   pFixedCoefs, Ipp32f* pSGain, IppsFullbandControllerState_EC_32f*
   pState);

IppStatus ippsFullbandController_EC_16s(const Ipp16s* pAdaptiveFilterErr,
   const Ipp16s* pFixedFilterErr, Ipp16s* pAdaptiveCoefs, Ipp16s*
   pFixedCoefs, Ipp32s* pSGain, IppsFullbandControllerState_EC_16s*
   pState);
```

FullbandControllerReset_EC

Resets fullband controller state.

```
IppStatus
   ippsFullbandControllerReset_EC_32f(IppsFullbandControllerState_EC_32f*
   pState);

IppStatus
   ippsFullbandControllerReset_EC_16s(IppsFullbandControllerState_EC_16s*
   pState);
```

FIR_EC

Computes FIR filter results.

```
IppStatus ippsFIR_EC_16s(const Ipp16s* pSrcSpchRef, const Ipp16s*
   pSrcSpch, Ipp16s* pDstSpch, int len, Ipp16s* pSrcTaps, Int tapsLen);

IppStatus ippsFIR_EC_32f(const Ipp32f* pSrcSpchRef, const Ipp32f*
   pSrcSpch, Ipp32f* pDstSpch, int len, Ipp32f* pSrcTaps, Int tapsLen);
```

NLMS_EC

Performs FIR filtering with coefficients update.

```
IppStatus ippsNLMS_EC_16s(const Ipp16s* pSrcSpchRef, const Ipp16s*
   pSrcSpch, const Ipp32s* pStepSize, Ipp16s* pSrcDstErr, Ipp16s*
   pDstSpch, int len, Ipp16s* pSrcDstTaps, int tapsLen);

IppStatus ippsNLMS_EC_32f(const Ipp32f* pSrcSpchRef, const Ipp32f*
   pSrcSpch, const Ipp32f* pStepSize, Ipp32f* pSrcDstErr,  Ipp32f*
   pDstSpch, int len, Ipp32f* pSrcDstTaps, int tapsLen);
```

G722 Sub-Band ADPCM Speech Codec Related Functions

SBADPCMEncodeStateSize_G722

Returns the memory size required for G.722 Sub-Band ADPCM encoder.

```
IppStatus ippsSBADPCMEncodeStateSize_G722_16s (int* pEncMemSize)
```

SBADPCMEncodeInit_G722

Initializes the memory buffer for the Sub-Band ADPCM encoder.

```
IppStatus ippsSBADPCMEncodeInit_G722_16s (IppsEncoderState_G722_16s*
   pEncMem)
```

SBADPCMEncode_G722

Performs Sub-Band ADPCM encoding of the synthesis QMF samples.

```
IppStatus ippsSBADPCMEncode_G722_16s (const Ipp16s* pSrc, Ipp16s* pDst,
   int len, IppsEncoderState_G722_16s* pEncMem)
```

QMFEncode_G722

Performs QMF analysis of the uniform PCM input.

```
IppStatus ippsQMFEncode_G722_16s(const Ipp16s* pSrc, Ipp16s* pDst, int
   len, Ipp16s* delay)
```

SBADPCMDecodeStateSize

Returns the memory size required for Sub-Band ADPCM decoder.

```
IppStatus ippsSBADPCMDecodeStateSize_G722 (int* pDecMemSize)
```

SBADPCMDecodeInit_G722

Initializes the memory buffer for the Sub-Band ADPCM decoder.

```
IppStatus ippsSBADPCMDecodeInit_G722_16s (IppsDecoderState_G722_16s*
   pDecMem)
```

SBADPCMDecode_G722

Performs decoding of the Sub-Band ADPCM compressed bit-stream.

```
IppStatus ippsSBADPCMDecode_G722_16s (const Ipp16s* pSrc, Ipp16s* pDst,
   int len, Ipp16s mode, IppsDecoderState_G722_16s* pDecMem)
```

QMFDecode_G722

Performs QMF synthesis of recovered samples.

```
IppStatus ippsQMFDecode_G722_16s (const Ipp16s* pSrc, Ipp16s* pDst, int
   len, Ipp16s* delay);
```

Companding Functions

MuLawToLin

Decodes samples from 8-bit m-law encoded format to linear samples.

```
IppStatus ippsMuLawToLin_8u16s(const Ipp8u* pSrc, Ipp16s* pDst, int len);
IppStatus ippsMuLawToLin_8u32f(const Ipp8u* pSrc, Ipp32f* pDst, int len);
```

LinToMuLaw
Encodes the linear samples using 8-bit m-law format and stores them in a vector.

```
IppStatus ippsLinToMuLaw_16s8u(const Ipp16s* pSrc, Ipp8u* pDst, int len);
IppStatus ippsLinToMuLaw_32f8u(const Ipp32f* pSrc, Ipp8u* pDst, int len);
```

ALawToLin
Decodes the 8-bit A-law encoded samples to linear samples.

```
IppStatus ippsALawToLin_8u16s(const Ipp8u* pSrc, Ipp16s* pDst, int len);
IppStatus ippsALawToLin_8u32f(const Ipp8u* pSrc, Ipp32f* pDst, int len);
```

LinToALaw
Encodes the linear samples using 8-bit A-law format and stores them in an array.

```
IppStatus ippsLinToALaw_16s8u(const Ipp16s* pSrc, Ipp8u* pDst, int len);
IppStatus ippsLinToALaw_32f8u(const Ipp32f* pSrc, Ipp8u* pDst, int len);
```

MuLawToALaw
Converts samples from 8-bit m-law encoded format to 8-bit A-law encoded format.

```
IppStatus ippsMuLawToALaw_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
```

ALawToMuLaw
Converts samples from 8-bit A-law encoded format to 8-bit m-law encoded format.

```
IppStatus ippsALawToMuLaw_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
```

Audio Coding Functions

Interleaved to Multi-Row Format Conversion Functions

Interleave
Converts signal from non-interleaved to interleaved format.

```
IppStatus ippsInterleave_16s(const Ipp16s** pSrc, int ch_num, int len,
   Ipp16s* pDst);
IppStatus ippsInterleave_32f(const Ipp32f** pSrc, int ch_num, int len,
   Ipp32f* pDst);
```

Deinterleave
Converts signal from interleaved to non-interleaved format.

```
IppStatus ippsDeinterleave_16s(const Ipp16s* pSrc, int ch_num, int len,
   Ipp16s** pDst);
IppStatus ippsDeinterleave_32f(const Ipp32f* pSrc, int ch_num, int len,
   Ipp32f** pDst);
```

Spectral Data Prequantization Functions

Pow34

Raises a vector to the power of 3/4.

```
IppStatus ippsPow34_32f16s(const Ipp32f* pSrc, Ipp16s* pDst, int len);
IppStatus ippsPow34_32f(const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsPow34_16s_Sfs(const Ipp16s* pSrc, int inscaleFactor, int
   Ipp16s* pDst, int scaleFactor, int len);
```

Pow43

Raises a vector to the power of 4/3.

```
IppStatus ippsPow43_16s32f(const Ipp16s* pSrc, Ipp32f* pDst, int len);
```

Scale Factors Calculation Functions

CalcSF

Restores actual scale factors from the bit stream values.

```
IppStatus ippsCalcSF_16s32f(const Ipp16s* pSrc, int offset, Ipp32s* pDst,
   int len);
```

Mantissa Conversion and Scaling Functions

Scale

Applies scale factors to spectral bands in accordance with spectral bands boundaries.

```
IppStatus ippsScale_32f_I(Ipp32f* pSrcDst, const Ipp32f* pSF,
   const int* pBandOffset, int bandsNumber);
```

MakeFloat

Converts mantissa and exponent arrays to float arrays.

```
IppStatus ippsMakeFloat_16s32f (Ipp32s* inmant, Ipp32s* inexp, Ipp32s
   size, Ipp32f* outfloat);
```

Modified Discrete Cosine Transform Functions

MDCTFwdInitAlloc, MDCTInvInitAlloc

Create and initialize modified discrete cosine transform specification structure.

```
IppStatus ippsMDCTFwdInitAlloc_32f(IppsMDCTFwdSpec_32f** ppMDCTSpec,
   int len);
IppStatus ippsMDCTFwdInitAlloc_16s(IppsMDCTFwdSpec_32s** ppMDCTSpec,
   int len);
```

```
IppStatus ippsMDCTInvInitAlloc_32f(IppsMDCTInvSpec_32f** ppMDCTSpec,
   int len);
```

MDCTFwdInit, MDCTInvInit

Initialize modified discrete cosine transform specification structure.

```
IppStatus ippsMDCTFwdInit_32f(IppsMDCTFwdSpec_32f** ppMDCTSpec, int len,
   Ipp8u* pMemSpec, Ipp8u* pMemInit);

IppStatus ippsMDCTFwdInit_16s(IppsMDCTFwdSpec_32f** ppMDCTSpec, int len,
   Ipp8u* pMemSpec, Ipp8u* pMemInit);

IppStatus ippsMDCTInvInit_32f(IppsMDCTInvSpec_32f** ppMDCTSpec, int len,
   Ipp8u* pMemSpec, Ipp8u* pMemInit);
```

MDCTFwdFree, MDCTInvFree

Closes modified discrete cosine transform specification structure.

```
IppStatus ippsMDCTFwdFree_32f(IppsMDCTFwdSpec_32f* pMDCTSpec);

IppStatus ippsMDCTFwdFree_16s(IppsMDCTFwdSpec_16s* pMDCTSpec);

IppStatus ippsMDCTInvFree_32f(IppsMDCTInvSpec_32f* pMDCTSpec);
```

MDCTFwdGetSize, MDCTInvGetSize

Get the sizes of MDCT specification structure, MDCT initialization, and MDCT work buffer.

```
IppStatus ippsMDCTFwdGetSize_32f(int len, int* pSizeSpec, int*
   pSizeInit, int* pSizeBuf);

IppStatus ippsMDCTFwdGetSize_16s(int len, int* pSizeSpec, int*
   pSizeInit, int* pSizeBuf);

IppStatus ippsMDCTInvGetSize_32f(int len, int* pSizeSpec, int*
   pSizeInit, int* pSizeBuf);
```

MDCTFwdGetBufSize, MDCTInvGetBufSize

Gets the size of MDCT work buffer.

```
IppStatus ippsMDCTFwdGetBufSize_32f(const IppsMDCTFwdSpec_32f*
   pMDCTSpec, int* pSize);

IppStatus ippsMDCTFwdGetBufSize_16s(const IppsMDCTFwdSpec_16s*
   pMDCTSpec, int* pSize);

IppStatus ippsMDCTInvGetBufSize_32f(const IppsMDCTInvSpec_32f*
   pMDCTSpec, int* pSize);
```

MDCTFwd, MDCTInv

Computes forward or inverse modified discrete cosine transform (MDCT) of a signal.

```
IppStatus ippsMDCTFwd_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsMDCTFwdSpec_32f* pMDCTSpec, Ipp8u* pBuffer);

IppStatus ippsMDCTInv_32f(const Ipp32f* pSrc, Ipp32f* pDst, const
   IppsMDCTInvSpec_32f* pMDCTSpec, Ipp8u* pBuffer);
```

```
IppStatus ippsMDCTFwd_32f_I(Ipp32f* pSrcDst, const IppsMDCTFwdSpec_32f*
   pMDCTSpec, Ipp8u* pBuffer);
IppStatus ippsMDCTInv_32f_I(Ipp32f* pSrcDst, const IppsMDCTInvSpec_32f*
   pMDCTSpec, Ipp8u* pBuffer);
IppStatus ippsMDCTFwd_16s_Sfs(const Ipp16s* pSrc, Ipp16s* pDst, const
   IppsMDCTFwdSpec_32f* pMDCTSpec, int scaleFactor, Ipp8u* pBuffer);
```

Block Filtering Functions

FIRBlockInitAlloc

Initializes FIR block filter state.

```
IppStatus ippsFIRBlockInitAlloc_32f(IppsFIRBlockState_32f** pState,
   int order, int len);
```

FIRBlockFree

Closes FIR block filter state.

```
IppStatus ippsFIRBlockFree_32f(IppsFIRBlockState_32f* pState);
```

FIRBlockOne

Filters vector of samples through FIR block filter.

```
IppStatus ippsFIRBlockOne_32f(Ipp32f* pSrc, Ipp32f* pDst,
   IppsFIRBlockState_32f* pState, Ipp32f* pTaps);
```

Frequency Domain Prediction Functions

FDPInitAlloc

Creates and initializes predictor state.

```
IppStatus ippsFDPInitAlloc_32f(IppsFDPState_32f** ppFDPState, int len);
```

FDPInit

Initializes predictor state.

```
IppStatus ippsFDPInit_32f(IppsFDPState_32f** ppFDPState, int len, Ipp8u*
   pMemSpec);
```

FDPFree

Closes FDP state.

```
IppStatus ippsFDPFree_32f(IppsFDPState_32f* pFDPState);
```

FDPGetSize

Gets size of FDP state structure.

```
IppStatus ippsFDPGetSize_32f(int len, int* pSizeState);
```

FDPReset
Resets predictors for all spectral lines.

```
IppStatus ippsFDPReset_32f(IppsFDPState_32f* pFDPState);
```

FDPResetSfb
Resets predictor-specific information in some scale factor bands.

```
IppStatus ippsFDPResetSfb_32f (const int* pSfbOffset, int sfbNumber,
   const Ipp8u* pResetFlag, IppsFDPState_32f* pFDPState);
```

FDPResetGroup
Resets predictors for group of spectral lines.

```
IppStatus ippsFDPResetGroup_32f (int resetGroupNumber, int step,
   IppsFDPState_32f* pFDPState);
```

FDPFwd
Performs frequency domain prediction procedure and calculates prediction error.

```
IppStatus ippsFDPFwd_32f(const Ipp32f* pSrc, Ipp32f* pDst,
   IppsFDPState_32f*  pFDPState);
```

FDPInv
Retrieves input signal from prediction error, using frequency domain prediction procedure.

```
IppStatus ippsFDPInv_32f_I(Ipp32f* pSrcDst, const int* pBandsOffset, int
   predictorBandsNumber, Ipp8u* pPredictionUsed, IppsFDPState_32f*
   pFDPState);
```

VLC Functions

VLCDecodeEscBlock_MP3
Parses the bitstream and decodes variable length code for MP3.

```
IppStatus ippsVLCDecodeEscBlock_MP3_1u16s(const Ipp8u** ppBitStream,
   int* pBitOffset, int linbits, Ipp16s* pData, int len, const
   IppsVLCDecodeSpec_32s* pVLCSpec);
```

VLCDecodeEscBlock_AAC
Parses the bitstream and decodes variable length code for AAC.

```
IppStatus ippsVLCDecodeEscBlock_AAC_1u16s(const Ipp8u** ppBitStream,
   int* pBitOffset, Ipp16s* pData, int len, const IppsVLCDecodeSpec_32s*
   pVLCSpec);
```

VLCCountEscBits_MP3
Calculates the number of bits necessary for encoding in MP3 format.

```
IppStatus ippsVLCCountEscBits_MP3_16s32s(const Ipp16s* pInputData, int
   len, int linbits, Ipp32s* pCountBits, const IppsVLCEncodeSpec_32s*
   pVLCSpec);
```

VLCCountEscBits_AAC
Calculates the number of bits necessary for encoding in AAC format.

```
IppStatus ippsVLCCountEscBits_AAC_16s32s(const Ipp16s* pInputData, int
   len, Ipp32s* pCountBits, const IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCEncodeEscBlock_MP3
Encodes an array of values into destination bitstream in MP3 format and advances bitstream pointer.

```
IppStatus ippsVLCEncodeEscBlock_MP3_16s1u(const Ipp16s* pInputData, int
   len, int linbits, Ipp8u** ppBitStream, int* pBitOffset, const
   IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCEncodeEscBlock_AAC
Encodes an array of values into destination bitstream in AAC format and advances bitstream pointer.

```
IppStatus ippsVLCEncodeEscBlock_AAC_16s1u(const Ipp16s* pInputData, int
   len, Ipp8u** ppBitStream, int* pBitOffset, const
   IppsVLCEncodeSpec_32s* pVLCSpec);
```

Psychoacoustic Functions

Spread
Computes spreading function.

```
IppStatus ippsSpread_16s_Sfs(Ipp32s* Src1, Ipp32s Src2, int
   inScaleFactor, Ipp32s* pDst);
```

Vector Quantization Functions

VQCodeBookInitAlloc
Creates and initializes the codebook structure.

```
IppStatus ippsVQCodeBookInitAlloc_32f(const Ipp32f* pInputTable,
   IppsVQCodeBookState_32f** ppCodeBook, int step, int height);
```

VQCodeBookInit
Initializes the codebook structure.

```
IppStatus ippsVQCodeBookInit_32f(const Ipp32f* pInputTable,
   IppsVQCodeBookState_32f** ppCodeBook, int step, int height, Ipp8u*
   pMemSpec);
```

VQCodeBookFree
Closes the `IppsVQCodeBookState_32f` structure created by the function `ippsVQCodeBookInitAlloc`.

```
IppStatus ippsVQCodeBookFree_32f(IppsVQCodeBookState_32f* pCodeBook);
```

VQCodeBookGetSize
Gets the size of the codebook structure.

```
IppStatus ippsVQCodeBookGetSize_32f(int step, int height, int*
   pSizeState);
```

VQPreliminarySelect
Selects candidates for the nearest code vector of codebooks.

```
IppStatus ippsVQPreliminarySelect_32f(const Ipp32f* pSrc, const Ipp32f*
   pWeights, int nDiv, const Ipp32s* pLengths, Ipp32s* pIndx, Ipps32s*
   pSign, int nCand, int* pPolbits, IppsVQCodeBookState_32f* pCodeBook);
```

VQMainSelect
Finds optimal indexes with minimal distortion.

```
IppStatus ippsVQMainSelect_32f(const Ipp32f* pSrc, const Ipp32f*
   pWeights, int nDiv, const Ipp32s* pLengths, int nCand, Ipp32s**
   ppIndexCand, Ipps32s** ppSignCand, Ipp32s** ppIndx, Ipp32s** ppSign,
   IppsVQCodeBookState_32f** ppCodeBooks, int nCodeBooks);
```

VQIndexSelect
Finds optimal vector set for specified number of codebooks.

```
IppStatus ippsVQIndexSelect_32f(const Ipp32f* pSrc, const Ipp32f*
   pWeights, int nDiv, const Ipp32s* pLengths,  int nCand, int**
   ppPolbits, Ipp32s** ppIndx, Ipp32s** ppSign,
   IppsVQCodeBookState_32f** ppCodeBooks, int nCodeBooks);
```

VQReconstruction
Reconstructs vectors from indexes.

```
IppStatus ippsVQReconstruction_32f(const Ipp32s** ppIndx, const Ipp32s**
   ppSign, const Ipp32s* pLengths, int nDiv, Ipp32f* pDst,
   IppsVQCodeBookState_32f** ppCodeBooks, int nCodeBooks);
```

MP3 Audio Coding Functions

AnalysisPQMF_MP3
Implements stage 1 of MP3 hybrid analysis filterbank.

```
IppStatus ippsAnalysisPQMF_MP3_16s32s(const Ipp16s* pSrcPcm,
   Ipp32s* pDstS, int pcmMode);
```

MDCTFwd_MP3
Implements stage 2 of the MP3 hybrid analysis filterbank.

```
IppStatus ippsMDCTFwd_MP3_32s(const Ipp32s* pSrc, Ipp32s* pDst, int
   blockType, int mixedBlock, IppMP3FrameHeader* pFrameHeader, Ipp32s*
   pOverlapBuf);
```

PsychoacousticModelTwo_MP3

Implements ISO/IEC 11172-3 psychoacoustic model recommendation 2 to estimate masked threshold and perceptual entropy associated with a block of PCM audio input.

```
IppStatus ippsPsychoacousticModelTwo_MP3_16s(const Ipp16s* pSrcPcm,
   IppMP3PsychoacousticModelTwoAnalysis* pDstPsyInfo, int*
   pDstIsSfbBound, IppMP3SideInfo* pDstSideInfo, IppMP3FrameHeader*
   pFrameHeader, IppMP3PsychoacousticModelTwoState* pFramePsyState,
   Ipp32s* pWorkBuffer,
   int pcmMode);
```

JointStereoEncode_MP3

Transforms independent left and right channel spectral coefficient vectors into combined mid/side and/or intensity mode coefficient vectors suitable for quantization.

```
IppStatus ippsJointStereoEncode_MP3_32s_I(Ipp32s* pSrcDstXrL, Ipp32s*
   pSrcDstXrR, Ipp8s* pDstScaleFactorR, IppMP3FrameHeader* pFrameHeader,
   IppMP3SideInfo* pSideInfo, int* pIsSfbBound);
```

Quantize_MP3

Quantizes spectral coefficients generated by analysis filterbank.

```
IppStatus ippsQuantize_MP3_32s_I(Ipp32s* pSrcDstXrIx, Ipp8s*
   pDstScalefactor, int* pDstScfsi, int* pDstCount1Len, int*
   pDstHufSize, IppMP3FrameHeader* pFrameHeader, IppMP3SideInfo*
   pSideInfo, IppMP3PsychoacousticModelTwoAnalysis* pPsychoInfo,
   IppMP3PsychoacousticModelTwoState* pFramePsyState,
   IppMP3BitReservoir* pResv, int meanBits, int* pIsSfbBound, Ipp32s*
   pWorkBuffer);
```

PackScalefactors_MP3

Applies noiseless coding to scalefactors and packs output into bitstream buffer.

```
IppStatus ippsPackScalefactors_MP3_8s1u(const Ipp8s* pSrcScalefactor,
   Ipp8u** ppBitStream, int* pOffset, IppMP3FrameHeader* pFrameHeader,
   IppMP3SideInfo* pSideInfo, int* pScfsi, int granule, int channel);
```

HuffmanEncode_MP3

Applies lossless Huffman encoding to quantized samples and packs output into bitstream buffer.

```
IppStatus ippsHuffmanEncode_MP3_32s1u(Ipp32s* pSrcIx, Ipp8u**
   ppBitStream, int* pOffset, IppMP3FrameHeader* pFrameHeader,
   IppMP3SideInfo* pSideInfo, int count1Len, int hufSize);
```

PackFrameHeader_MP3

Packs the content of the frame header into the bitstream.

```
IppStatus ippsPackFrameHeader_MP3 (IppMP3FrameHeader* pSrcFrameHeader,
   Ipp8u** ppBitStream);
```

PackSideInfo_MP3

Packs the side information into the bitstream buffer.

```
IppStatus ippsPackSideInfo_MP3(IppMP3SideInfo* pSrcSideInfo, Ipp8u**
   ppBitStream, int mainDataBegin, int privateBits, int* pSrcScfsi,
   IppMP3FrameHeader* pFrameHeader);
```

BitReservoirInit_MP3

Initializes all elements of the bit reservoir state structure.

```
IppStatus ippsBitReservoirInit_MP3(IppMP3BitReservoir* pDstBitResv,
   IppMP3FrameHeader* pFrameHeader);
```

UnpackFrameHeader_MP3

Unpacks audio frame header.

```
IppStatus ippsUnpackFrameHeader_MP3(Ipp8u** ppBitStream,
   IppMP3FrameHeader* pFrameHeader);
```

UnpackSideInfo_MP3

Unpacks side information from input bitstream for use during decoding of associated frame.

```
IppStatus ippsUnpackSideInfo_MP3(Ipp8u** ppBitStream, IppMP3SideInfo*
   pDstSideInfo, int* pDstMainDataBegin, int* pDstPrivateBits, int*
   pDstScfsi,  IppMP3FrameHeader* pFrameHeader);
```

UnpackScaleFactors_MP3

Unpacks scalefactors.

```
IppStatus ippsUnpackScaleFactors_MP3_1u8s(Ipp8u** ppBitStream, int*
   pOffset, Ipp8s* pDstScaleFactor, IppMP3SideInfo* pSideInfo, int*
   pScfsi, IppMP3FrameHeader* pFrameHeader, int granule, int channel);
```

HuffmanDecode_MP3
HuffmanDecodeSfb_MP3
HuffmanDecodeSfbMbp_MP3

Decodes Huffman data.

```
IppStatus ippsHuffmanDecode_MP3_1u32s(Ipp8u** ppBitStream, int* pOffset,
   Ipp32s* pDstIs, int* pDstNonZeroBound, IppMP3SideInfo* pSideInfo,
   IppMP3FrameHeader* pFrameHeader, int hufSize);
```

```
IppStatus ippsHuffmanDecodeSfb_MP3_1u32s(Ipp8u** ppBitStream, int*
   pOffset, Ipp32s* pDstIs, int* pDstNonZeroBound, IppMP3SideInfo*
   pSideInfo, IppMP3FrameHeader* pFrameHeader, int hufSize,
   IppMP3ScaleFactorBandTableLong* pSfbTableLong);
```

```
IppStatus, ippsHuffmanDecodeSfbMbp_MP3_1u32s(Ipp8u** ppBitStream, int*
   pOffset, Ipp32s* pDstIs, int* pDstNonZeroBound, IppMP3SideInfo*
   pSideInfo, IppMP3FrameHeader* pFrameHeader, int hufSize,
   IppMP3ScaleFactorBandTableLong* pSfbTableLong,
   IppMP3ScaleFactorBandTableShort pSfbTableShort,
   IppMP3MixedBlockPartitionTable pMbpTable);
```

ReQuantize_MP3
ReQuantizeSfb_MP3
Requantizes the decoded Huffman symbols.

```
IppStatus ippsReQuantize_MP3_32s_I(Ipp32s* pSrcDstIsXr, int*
   pNonZeroBound, Ipp8s* pScaleFactor, IppMP3SideInfo* pSideInfo,
   IppMP3FrameHeader* pFrameHeader, Ipp32s* pBuffer);
IppStatus ippsReQuantizeSfb_MP3_32s_I(Ipp32s* pSrcDstIsXr, int*
   pNonZeroBound, Ipp8s* pScaleFactor, IppMP3SideInfo* pSideInfo,
   IppMP3FrameHeader* pFrameHeader, Ipp32s* pBuffer,
   IppMP3ScaleFactorBandTableLong pSfbTableLong,
   IppMP3ScaleFactorBandTableShort pSfbTableShort);
```

MDCTInv_MP3
Performs the first stage of hybrid synthesis filter bank.

```
IppStatus ippsMDCTInv_MP3_32s(Ipp32s* pSrcXr, Ipp32s* pDstY,  Ipp32s*
   pSrcDstOverlapAdd, int nonZeroBound,  int* pPrevNumOfImdct, int
   blockType, int mixedBlock);
```

SynthPQMF_MP3
Performs the second stage of hybrid synthesis filter bank.

```
IppStatus ippsSynthPQMF_MP3_32s16s(Ipp32s* pSrcY, Ipp16s* pDstAudioOut,
   Ipp32s* pVBuffer, int* pVPosition, int mode);
```

Advanced Audio Coding Functions

UnpackADIFHeader_AAC
Gets the AAC ADIF format header.

```
IppStatus ippsUnpackADIFHeader_AAC(Ipp8u** ppBitStream,
   IppAACADIFHeader* pADIFHeader, IppAACPrgCfgElt* pPrgCfgElt, int
   prgCfgEltMax);
```

UnpackADTSFrameHeader_AAC
Gets ADTS frame header from the input bitstream.

```
IppStatus ippsUnpackADTSFrameHeader_AAC(Ipp8u** ppBitStream,
   IppAACADTSFrameHeader* pADTSFrameHeader);
```

DecodePrgCfgElt_AAC
Gets program configuration element from the input bitstream.

```
IppStatus ippsDecodePrgCfgElt_AAC(Ipp8u** ppBitStream, int* pOffset,
   IppAACPrgCfgElt* pPrgCfgElt);
```

DecodeChanPairElt_AAC
Gets `channel_pair_element` from the input bitstream.

```
IppStatus ippsDecodeChanPairElt_AAC(Ipp8u** ppBitStream, int* pOffset,
   IppAACIcsInfo* pIcsInfo, IppAACChanPairElt* pChanPairElt, int
   predSfbMax);
```

NoiselessDecoder_LC_AAC

Decodes all data for one channel.

```
IppStatus ippsNoiselessDecoder_LC_AAC(Ipp8u** ppBitStream, int* pOffset,
   int commonWin, IppAACChanInfo* pChanInfo, Ipp16s* pDstScalefactor,
   Ipp32s* pDstQuantizedSpectralCoef, Ipp8u* pDstSfbCb, Ipp8s*
   pDstTnsFiltCoef);
```

DecodeDatStrElt_AAC

Gets data stream element from the input bitstream.

```
IppStatus ippsDecodeDatStrElt_AAC(Ipp8u** ppBitStream, int* pOffset,
   int* pDataTag, int* pDataCnt, Ipp8u* pDstDataElt);
```

DecodeFillElt_AAC

Gets the fill element from the input bitstream.

```
IppStatus ippsDecodeFillElt_AAC(Ipp8u** ppBitStream, int* pOffset, int*
   pFillCnt, Ipp8u* pDstFillElt);
```

QuantInv_AAC

Performs inverse quantization of Huffman symbols for current channel in-place.

```
IppStatus ippsQuantInv_AAC_32s_I(Ipp32s* pSrcDstSpectralCoef, const
   Ipp16s* pScalefactor, int numWinGrp, const int* pWinGrpLen, int
   maxSfb, const Ipp8u* pSfbCb, int samplingRateIndex, int winLen);
```

DecodeMsStereo_AAC

Processes mid-side (MS) stereo for pair channels in-place.

```
IppStatus ippsDecodeMsStereo_AAC_32s_I(Ipp32s* pSrcDstL, Ipp32s*
   pSrcDstR, int msMaskPres, const Ipp8u* pMsUsed, Ipp8u* pSfbCb, int
   numWinGrp, const int* pWinGrpLen, int maxSfb, int samplingRateIndex,
   int winLen);
```

DecodeIsStereo_AAC

Processes intensity stereo for pair channels.

```
IppStatus ippsDecodeIsStereo_AAC_32s(const Ipp32s* pSrcL, Ipp32s* pDstR,
   const Ipp16s* pScalefactor, const Ipp8u* pSfbCb, int numWinGrp,
   const int* pWinGrpLen, int maxSfb, int samplingRateIndex, int
   winLen);
```

DeinterleaveSpectrum_AAC

Deinterleaves the coefficients for short block.

```
IppStatus ippsDeinterleaveSpectrum_AAC_32s(const Ipp32s* pSrc, Ipp32s*
   pDst, int numWinGrp, const int* pWinGrpLen, int maxSfb, int
   samplingRateIndex, int winLen);
```

DecodeTNS_AAC

Decodes for Temporal Noise Shaping in-place.

```
IppStatus ippsDecodeTNS_AAC_32s_I(Ipp32s* pSrcDstSpectralCoefs, const
   int* pTnsNumFilt, const int* pTnsRegionLen, const int* pTnsFiltOrder,
   const int* pTnsFiltCoefRes, const Ipp8s* pTnsFiltCoef, const int*
   pTnsDirection, int maxSfb, int profile, int samplingRateIndex, int
   winLen);
```

MDCTInv_AAC

Maps time-frequency domain signal into time domain and generates 1024 reconstructed 16-bit signed little-endian PCM samples.

```
IppStatus ippsMDCTInv_AAC_32s16s(Ipp32s* pSrcSpectralCoefs, Ipp16s*
   pDstPcmAudioOut, Ipp32s* pSrcDstOverlapAddBuf, int winSequence, int
   winShape, int prevWinShape, int pcmMode);
```

DecodeMainHeader_AAC

Gets main header information and main layer information from bit stream.

```
IppStatus ippsDecodeMainHeader_AAC(Ipp8u** ppBitStream, int* pOffset,
   IppAACMainHeader* pAACMainHeader, int channelNum, int
   monoStereoFlag);
```

DecodeExtensionHeader_AAC

Gets extension header information and extension layer information from bit stream.

```
IppStatus ippsDecodeExtensionHeader_AAC(Ipp8u** ppBitStream, int*
   pOffset, IppAACExtHeader* pAACExtHeader, int monoStereoFlag, int
   thisLayerStereo, int monoLayerFlag, int preStereoMaxSfb, int
   hightstMonoMaxSfb, int winSequence);
```

DecodePNS_AAC

Implements perceptual noise substitution (PNS) coding within individual channel stream (ICS).

```
IppStatus ippsDecodePNS_AAC_32s(Ipp32s* pSrcDstSpec, int*
   pSrcDstLtpFlag, Ipp8u* pSfbCb, Ipp16s* pScaleFactor, int maxSfb, int
   numWinGrp, int* pWinGrpLen, int samplingFreqIndex, int winLen, int*
   pRandomSeed);
```

DecodeMsPNS_AAC

Implements perceptual noise substitution (PNS) coding in the case of joint coding.

```
IppStatus ippsDecodeMsPNS_AAC_32s(Ipp32s* pSrcDstSpec, int*
   pSrcDstLtpFlag, Ipp8u* pSfbCb, Ipp16s* pScaleFactor, int maxSfb, int
   numWinGrp, int* pWinGrpLen, int samplingFreqIndex, int winLen, int*
   pRandomSeed, int channel, Ipp8u* pMsUsed, int* pNoiseState);
```

DecodeChanPairElt_MP4_AAC

Gets `channel_pair_element` from the input bitstream.

```
IppStatus ippsDecodeChanPairElt_MP4_AAC(Ipp8u** ppBitStream, int*
   pOffset, IppAACIcsInfo* pIcsInfo, IppAACChanPairElt* pChanPairElt,
   IppAACMainHeader* pAACMainHeader, int predSfbMax, int
   audioObjectType);
```

LongTermReconstruct_AAC
Uses Long Term Reconstruct (LTR) to reduce signal redundancy between successive coding frames.

```
IppStatus ippsLongTermReconstruct_AAC_32s(Ipp32s* pSrcEstSpec, Ipp32s*
   pSrcDstSpec, int* pLtpFlag, int winSequence, int samplingFreqIndex);
```

MDCTFwd_AAC
Generates spectrum coefficient of PCM samples.

```
IppStatus ippsMDCTFwd_AAC_32s(Ipp32s* pSrc, Ipp32s* pDst, Ipp32s*
   pSrcDstOverlapAdd, int winSequence, int winShape, int preWinShape,
   Ipp32s* pWindowedBuf);
```

EncodeTNS_AAC
Performs reversion of TNS in the Long Term Reconstruct loop in-place.

```
IppStatus ippsEncodeTNS_AAC_32s_I(Ipp32s* pSrcDst, const int*
   pTnsNumFilt, const int* pTnsRegionLen, const int* pTnsFiltOrder,
   const int* pTnsFiltCoefRes, const Ipp8s* pTnsFiltCoef, const int*
   pTnsDirection, int maxSfb, int profile, int samplingRateIndex, int
   winLen);
```

LongTermPredict_AAC
Gets the predicted time domain signals in the Long Term Reconstruct (LTP) loop.

```
IppStatus ippsLongTermPredict_AAC_32s(Ipp32s* pSrcTimeSignal, Ipp32s*
   pDstEstTimeSignal, IppAACLtpInfo* pAACLtpInfo, int winSequence);
```

NoiselessDecode_AAC
Performs noiseless decoding.

```
IppStatus ippsNoiselessDecode_AAC(Ipp8u** ppBitStream, int* pOffset,
   IppAACMainHeader* pAACMainHeader, Ipp16s* pDstScaleFactor, Ipp32s*
   pDstQuantizedSpectralCoef, Ipp8u* pDstSfbCb, Ipp8s* pDstTnsFiltCoef,
   IppAACChanInfo* pChanInfo, int winSequence, int maxSfb, int
   commonWin, int scaleFlag, int audioObjectType);
```

LtpUpdate_AAC
Performs required buffer update in the Long Term Reconstruct (LTP) loop.

```
IppStatus ippsLtpUpdate_AAC_32s(Ipp32s* pSpecVal, Ipp32s* pLtpSaveBuf,
   int winSequence, int winShape, int preWinShape, Ipp32s* pWorkBuf);
```

Spectral Band Replication Functions

AnalysisFilterGetSize_SBR
Returns size of `FilterSpec_SBR` specification structures, init and work buffers.

```
IppStatus ippsAnalysisFilterGetSize_SBR_RToC_32f32fc(int* pSizeSpec,
   int* pSizeInitBuf, int* pSizeWorkBuf);
IppStatus ippsAnalysisFilterGetSize_SBR_RToC_32f(int* pSizeSpec, int*
   pSizeInitBuf, int* pSizeWorkBuf);
```

```
IppStatus ippsAnalysisFilterGetSize_SBR_RToR_32f(int* pSizeSpec, int*
   pSizeInitBuf, int* pSizeWorkBuf);
```

SynthesisFilterGetSize_SBR

Returns size of `FilterSpec_SBR` specification structures, init and work buffers.

```
IppStatus ippsSynthesisFilterGetSize_SBR_CToR_32fc32f(int* pSizeSpec,
   int* pSizeInitBuf, int* pSizeWorkBuf);
IppStatus ippsSynthesisFilterGetSize_SBR_CToR_32f(int* pSizeSpec, int*
   pSizeInitBuf, int* pSizeWorkBuf);
IppStatus ippsSynthesisFilterGetSize_SBR_RToR_32f(int* pSizeSpec, int*
   pSizeInitBuf, int* pSizeWorkBuf);
```

SynthesisDownFilterGetSize_SBR

Returns size of `FilterSpec_SBR` specification structures, init and work buffers.

```
IppStatus ippsSynthesisDownFilterGetSize_SBR_CToR_32fc32f(int*
   pSizeSpec, int* pSizeInitBuf, int* pSizeWorkBuf);
IppStatus ippsSynthesisDownFilterGetSize_SBR_CToR_32f(int* pSizeSpec,
   int* pSizeInitBuf, int* pSizeWorkBuf);
IppStatus ippsSynthesisDownFilterGetSize_SBR_RToR_32f(int* pSizeSpec,
   int* pSizeInitBuf, int* pSizeWorkBuf);
```

AnalysisFilterInit_SBR

Initializes analysis specification structure.

```
IppStatus
   ippsAnalysisFilterInit_SBR_RToC_32f32fc(IppsFilterSpec_SBR_C_32fc**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus ippsAnalysisFilterInit_SBR_RToC_32f(IppsFilterSpec_SBR_C_32f**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus ippsAnalysisFilterInit_SBR_RToR_32f(IppsFilterSpec_SBR_R_32f**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
```

SynthesisFilterInit_SBR

Initializes synthesis specification structure.

```
IppStatus
   ippsSynthesisFilterInit_SBR_CToR_32fc32f(IppsFilterSpec_SBR_C_32fc**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus
   ippsSynthesisFilterInit_SBR_CToR_32f(IppsFilterSpec_SBR_C_32f**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus
   ippsSynthesisFilterInit_SBR_RToR_32f(IppsFilterSpec_SBR_R_32f**
   ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
```

SynthesisDownFilterInit_SBR

Initializes synthesis down specification structure.

```
IppStatus
    ippsSynthesisDownFilterInit_SBR_CToR_32fc32f(IppsFilterSpec_SBR_C_32
    fc** ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus
    ippsSynthesisDownFilterInit_SBR_CToR_32f(IppsFilterSpec_SBR_C_32f**
    ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
IppStatus
    ippsSynthesisDownFilterInit_SBR_RToR_32f(IppsFilterSpec_SBR_R_32f**
    ppFilterSpec, Ipp8u* pMemSpec, Ipp8u* pInitBuf);
```

AnalysisFilter_SBR

Splits time domain signal output from the core decoder into 32 subband signals.

```
IppStatus ippsAnalysisFilter_SBR_RToC_32f32fc_D2L(const Ipp32f* pSrc,
    Ipp32fc* pDst[], const Ipp32f* pSbrTableWindowDown, int NumLoop, int
    offset, int kx, const IppsFilterSpec_SBR_C_32fc* pFilterSpec, Ipp8u*
    pWorkBuf);
IppStatus ippsAnalysisFilter_SBR_RToC_32f_D2L(const Ipp32f* pSrc,
    Ipp32f* pDstRe[], Ipp32f* pDstIm[], const Ipp32f*
    pSbrTableWindowDown, int NumLoop, int offset, int kx, const
    IppsFilterSpec_SBR_C_32f* pFilterSpec, Ipp8u* pWorkBuf);
IppStatus ippsAnalysisFilter_SBR_RToR_32f_D2L(const Ipp32f* pSrc,
    Ipp32f* pDst[], const Ipp32f* pSbrTableWindowDown, int NumLoop, int
    offset, int kx, const IppsFilterSpec_SBR_R_32f* pFilterSpec, Ipp8u*
    pWorkBuf);
```

SynthesisFilter_SBR

Transforms SBR-processed subband signals into time domain samples.

```
IppStatus ippsSynthesisFilter_SBR_CToR_32fc32f_D2L (const Ipp32fc*
    pSrc[], Ipp32f* pDst, const Ipp32f* pSbrTableWindow, int NumLoop,
    const IppsFilterSpec_SBR_C_32fc* pFilterSpec, Ipp8u* pWorkBuf);
IppStatus ippsSynthesisFilter_SBR_CToR_32f_D2L(const Ipp32f* pSrcRe[],
    const Ipp32f* pSrcIm[], Ipp32f* pDst, const Ipp32f* pSbrTableWindow,
    int NumLoop, const IppsFilterSpec_SBR_C_32f* pFilterSpec, Ipp8u*
    pWorkBuf);
IppStatus ippsSynthesisFilter_SBR_RToR_32f_D2L(const Ipp32f* pSrc[],
    Ipp32f* pDst, const Ipp32f* pSbrTableWindow, int NumLoop, const
    IppsFilterSpec_SBR_R_32f* pFilterSpec, Ipp8u* pWorkBuf);
```

SynthesisDownFilter_SBR

Transforms SBR-processed subband signals into time domain samples and performs downsampling at the same time.

```
IppStatus ippsSynthesisDownFilter_SBR_CToR_32fc32f_D2L (const Ipp32fc*
    pSrc[], Ipp32f* pDst, const Ipp32f* pSbrTableWindowDown, const int
    NumLoop, const IppsFilterSpec_SBR_C_32fc* pFilterSpec, Ipp8u*
    pWorkBuf);
```

```
IppStatus ippsSynthesisDownFilter_SBR_CToR_32f_D2L(const Ipp32f*
   pSrcRe[], const Ipp32f* pSrcIm[], Ipp32f* pDst, const Ipp32f*
   pSbrTableWindow, const int NumLoop, const IppsFilterSpec_SBR_C_32f*
   pFilterSpec, Ipp8u* pWorkBuf);
IppStatus ippsSynthesisDownFilter_SBR_RToR_32f_D2L(const Ipp32f* pSrc[],
   Ipp32f* pDst, const Ipp32f* pSbrTableWindow, const int NumLoop, const
   IppsFilterSpec_SBR_R_32f* pFilterSpec, Ipp8u* pWorkBuf);
```

PredictCoef_SBR

Obtains prediction filter coefficients using covariance method.

```
IppStatus ippsPredictCoef_SBR_C_32fc_D2L(const Ipp32fc* pSrc[], Ipp32fc*
   pAlpha0, Ipp32fc* pAlpha1, int k0, int len);
IppStatus ippsPredictCoef_SBR_C_32f_D2L(const Ipp32f* pSrcRe[], const
   Ipp32f* pSrcIm[], Ipp32f* pAlpha0Re, Ipp32f* pAlpha0Im, Ipp32f*
   pAlpha1Re, Ipp32f* pAlpha1Im, int k0, int len);
IppStatus ippsPredictCoef_SBR_R_32f_D2L(const Ipp32f* pSrc[], Ipp32f*
   pAlpha0, Ipp32f* pAlpha1, int k0, int len);
```

String Functions

String Manipulation

Find, FindRev

Looks for the first occurrence of the substring matching the specified string.

```
IppStatus ippsFind_8u(const Ipp8u* pSrc, int len, const Ipp8u* pFind, int
   lenFind, int* pIndex);
IppStatus ippsFind_16u(const Ipp16u* pSrc, int len, const Ipp16u* pFind,
   int lenFind, int* pIndex);
IppStatus ippsFindRev_8u(const Ipp8u* pSrc, int len, const Ipp8u* pFind,
   int lenFind, int* pIndex);
IppStatus ippsFindRev_16u(const Ipp16u* pSrc, int len, const Ipp16u*
   pFind, int lenFind, int* pIndex);
```

FindC, FindRevC

Looks for the first occurrence of the specified element within the source string.

```
IppStatus ippsFindC_8u(const Ipp8u* pSrc, int len, Ipp8u valFind, int*
   pIndex);
IppStatus ippsFindC_16u(const Ipp16u* pSrc, int len, Ipp16u valFind, int*
   pIndex);
IppStatus ippsFindRevC_8u(const Ipp8u* pSrc, int len, Ipp8u valFind, int*
   pIndex);
```

```
IppStatus ippsFindRevC_16u(const Ipp16u* pSrc, int len, Ipp16u valFind,
   int* pIndex);
```

FindCAny, FindRevCAny

Looks for the first occurrence of any element of the specified array within the source string.

```
IppStatus ippsFindCAny_8u(const Ipp8u* pSrc, int len, const Ipp8u*
   pAnyOf, int lenAnyOf, int* pIndex);

IppStatus ippsFindCAny_16u(const Ipp16u* pSrc, int len, const Ipp16u*
   pAnyOf, int lenAnyOf, int* pIndex);

IppStatus ippsFindRevCAny_8u(const Ipp8u* pSrc, int len,
   const Ipp8u* pAnyOf, int lenAnyOf, int* pIndex);

IppStatus ippsFindRevCAny_16u(const Ipp16u* pSrc, int len, const Ipp16u*
   pAnyOf, int lenAnyOf, int* pIndex);
```

Insert

Inserts a string into another string.

```
IppStatus ippsInsert_8u(const Ipp8u* pSrc, int srcLen, const Ipp8u*
   pInsert, int insertLen, Ipp8u* pDst, int startIndex);

IppStatus ippsInsert_16u(const Ipp16u* pSrc, int srcLen, const Ipp16u*
   pInsert, int insertLen, Ipp16u* pDst, int startIndex);

IppStatus ippsInsert_8u_I(const Ipp8u* pInsert, int insertLen, Ipp8u*
   pSrcDst, int* pSrcDstLen, int startIndex);

IppStatus ippsInsert_16u_I(const Ipp16u* pInsert, int insertLen, Ipp16u*
   pSrcDst, int* pSrcDstLen, int startIndex);
```

Remove

Removes a specified number of elements from the string.

```
IppStatus ippsRemove_8u(const Ipp8u* pSrc, int srcLen, Ipp8u* pDst, int
   startIndex, int len);

IppStatus ippsRemove_16u(const Ipp16u* pSrc, int srcLen, Ipp16u* pDst,
   int startIndex, int len);

IppStatus ippsRemove_8u_I(Ipp8u* pSrcDst, int* pSrcDstLen, int
   startIndex, int len);

IppStatus ippsRemove_16u_I(Ipp16u* pSrcDst, int* pSrcDstLen, int
   startIndex, int len);
```

Compare

Compares two strings of the fixed length.

```
IppStatus ippsCompare_8u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, int len,
   int* pResult);

IppStatus ippsCompare_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, int
   len, int* pResult);
```

CompareIgnoreCase, CompareIgnoreCaseLatin

Compares two strings of the fixed length ignoring case.

```
IppStatus ippsCompareIgnoreCase_16u(const Ipp16u* pSrc1, const Ipp16u*
   pSrc2, int len, int* pResult);
IppStatus ippsCompareIgnoreCaseLatin_8u(const Ipp8u* pSrc1, const Ipp8u*
   pSrc2, int len, int* pResult);
IppStatus ippsCompareIgnoreCaseLatin_16u(const Ipp16u* pSrc1, const
   Ipp16u* pSrc2, int len, int* pResult);
```

Equal

Compares two string of the fixed length for equality.

```
IppStatus ippsEqual_8u(const Ipp8u* pSrc1, const Ipp8u* pSrc2, int len,
   int* pResult);
IppStatus ippsEqual_16u(const Ipp16u* pSrc1, const Ipp16u* pSrc2, int
   len, int* pResult);
```

TrimC

Deletes all occurrences of a specified symbol in the beginning and in the end of the string.

```
IppStatus ippsTrimC_8u(const Ipp8u* pSrc, int srcLen, Ipp8u odd, Ipp8u*
   pDst, int* pDstLen);
IppStatus ippsTrimC_16u(const Ipp16u* pSrc, int srcLen, Ipp16u odd,
   Ipp16u* pDst, int* pDstLen);
IppStatus ippsTrimC_8u_I(Ipp8u* pSrcDst, int* pLen, Ipp8u odd);
IppStatus ippsTrimC_16u_I(Ipp16u* pSrcDst, int* pLen, Ipp16u odd);
```

TrimCAny TrimStartCAny TrimEndCAny

Deletes all occurrences of any of the specified symbols in the beginning and in the end of the source string.

```
IppStatus ippsTrimCAny_8u(const Ipp8u* pSrc, int srcLen, const Ipp8u*
   pTrim, int trimLen, Ipp8u* pDst, int* pDstLen);
IppStatus ippsTrimCAny_16u(const Ipp16u* pSrc, int srcLen, const Ipp16u*
   pTrim, int trimLen, Ipp16u* pDst, int* pDstLen);
IppStatus ippsTrimStartCAny_8u(const Ipp8u* pSrc, int srcLen, const
   Ipp8u* pTrim, int trimLen, Ipp8u* pDst, int* pDstLen);
IppStatus ippsTrimStartCAny_16u(const Ipp16u* pSrc, int srcLen, const
   Ipp16u* pTrim, int trimLen, Ipp16u* pDst, int* pDstLen);
IppStatus ippsTrimEndCAny_8u(const Ipp8u* pSrc, int srcLen, const Ipp8u*
   pTrim, int trimLen, Ipp8u* pDst, int* pDstLen);
IppStatus ippsTrimEndCAny_16u(const Ipp16u* pSrc, int srcLen, const
   Ipp16u* pTrim, int trimLen, Ipp16u* pDst, int* pDstLen);
```

ReplaceC

Replaces all occurrences of a specified element in the source string with another element.

```
IppStatus ippsReplaceC_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len, Ipp8u
   oldVal, Ipp8u newVal);
IppStatus ippsReplaceC_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len,
   Ipp16u oldVal, Ipp16u newVal);
```

Uppercase, UppercaseLatin

Converts alphabetic characters of a string to all uppercase symbols.

```
IppStatus ippsUppercase_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len);
IppStatus ippsUppercase_16u_I(Ipp16u* pSrcDst, int len);
IppStatus ippsUppercaseLatin_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsUppercaseLatin_16u(const Ipp16u* pSrc, Ipp16u* pDst, int
   len);
IppStatus ippsUppercaseLatin_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsUppercaseLatin_16u_I(Ipp16u* pSrcDst, int len);
```

Lowercase, LowercaseLatin

Converts alphabetic characters of a string to all lowercase symbols.

```
IppStatus ippsLowercase_16u(const Ipp16u* pSrc, Ipp16u* pDst, int len);
IppStatus ippsLowercase_16u_I(Ipp16u* pSrcDst, int len);
IppStatus ippsLowercaseLatin_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len);
IppStatus ippsLowercaseLatin_16u(const Ipp16u* pSrc, Ipp16u* pDst, int
   len);
IppStatus ippsLowercaseLatin_8u_I(Ipp8u* pSrcDst, int len);
IppStatus ippsLowercaseLatin_16u_I(Ipp16u* pSrcDst, int len);
```

Hash

Calculates the hash value for the string.

```
IppStatus ippsHash_8u32u(const Ipp8u* pSrc, int len, Ipp32u* pHashVal);
IppStatus ippsHash_16u32u(const Ipp16u* pSrc, int len, Ipp32u* pHashVal);
```

Concat

Concatenates several strings together.

```
IppStatus ippsConcat_8u_D2L(const Ipp8u* const pSrc[], const int*
   pSrcLen[], int numSrc, Ipp8u* pDst);
IppStatus ippsConcat_16u_D2L(const Ipp16u* const pSrc[], const int*
   pSrcLen[], int numSrc, Ipp16u* pDst);
```

```
IppStatus ippsConcat_8u(const Ipp8u* pSrc1, int len1, const Ipp8u* pSrc2,
   int len2, Ipp8u* pDst);
IppStatus ippsConcat_16u(const Ipp16u* pSrc1, int len1, const Ipp16u*
   pSrc2, int len2, Ipp16u* pDst);
```

ConcatC

Concatenates several strings together and inserts symbol delimiters between them.

```
IppStatus ippsConcatC_8u_D2L(const Ipp8u* const pSrc[], const int*
   pSrcLen[], int numSrc, Ipp8u delim, Ipp8u* pDst);
IppStatus ippsConcatC_16u_D2L(const Ipp16u* const pSrc[], const int*
   pSrcLen[], int numSrc, Ipp16u delim, Ipp16u* pDst);
```

SplitC

Splits source string into separate parts.

```
IppStatus ippsSplitC_8u_D2L(const Ipp8u* pSrc, int srcLen, Ipp8u delim,
   Ipp8u* pDst[], int* pDstLen[], int* pNumDst);
IppStatus ippsSplitC_16u_D2L(const Ipp16u* pSrc, int srcLen, Ipp16u
   delim, Ipp16u* pDst[], int* pDstLen[], int* pNumDst);
```

Regular Expressions

RegExpInitAlloc

Allocates memory and initializes the structure for processing matching operation with regular expressions.

```
IppStatus ippsRegExpInitAlloc(const char* pPattern, const char*
   pOptions, IppRegExpState** ppRegExpState, int* pErrOffset);
```

RegExpFree

Frees the memory allocated for a regular expression state structure.

```
IppStatus ippsRegExpFree(IppsRegExpState* pRegExpState);
```

RegExpInit

Initializes the structure for processing matching operation with regular expressions.

```
IppStatus ippsRegExpInit(const char* pPattern, const char* pOptions,
   IppRegExpState* pRegExpState, int* pErrOffset)
```

RegExpGetSize

Computes the size of the regular expression state structure.

```
IppStatus ippsRegExpGetSize(const char* pPattern, int*
   pRegExpStateSize);
```

RegExpSetMatchLimit

Sets the value of the `matchLimit` parameter.

```
IppStatus ippsRegExpSetMatchLimit(int matchLimit, IppRegExpState*
   pRegExpState);
```

RegExpFind

Looks for the occurrences of the substrings matching the specified regular expression.

```
IppStatus ippsRegExpFind_8u(const Ipp8u* pSrc, int srcLen,
   IppRegExpState* pRegExpState, IppRegExpFind* pFind, int* pNumFind);
```

Fixed-Accuracy Arithmetic Functions

Power and Root Functions

Inv

Computes inverse value of each vector element.

```
IppStatus ippsInv_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInv_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInv_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInv_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsInv_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Div

Divides each element of the first vector by corresponding element of the second vector.

```
IppStatus ippsDiv_32f_A11 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsDiv_32f_A21 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f*  pDst, int len);
IppStatus ippsDiv_32f_A24 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f*  pDst, int len);
IppStatus ippsDiv_64f_A50 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
IppStatus ippsDiv_64f_A53 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
```

Sqrt

Computes square root of each vector element.

```
IppStatus ippsSqrt_32f_A11 ( const Ipp32f*  pSrc, Ipp32f*  pDst, int len
   );
IppStatus ippsSqrt_32f_A21 ( const Ipp32f*  pSrc, Ipp32f*  pDst, int len
   );
IppStatus ippsSqrt_32f_A24 ( const Ipp32f*  pSrc, Ipp32f*  pDst, int len
   );
IppStatus ippsSqrt_64f_A50 ( const Ipp64f*  pSrc, Ipp64f*  pDst, int len
   );
IppStatus ippsSqrt_64f_A53 ( const Ipp64f*  pSrc, Ipp64f*  pDst, int len
   );
```

InvSqrt

Computes inverse square root of each vector element.

```
IppStatus ippsInvSqrt_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvSqrt_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvSqrt_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvSqrt_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsInvSqrt_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Cbrt

Computes cube root of each vector element.

```
IppStatus ippsCbrt_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCbrt_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCbrt_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCbrt_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsCbrt_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

InvCbrt

Computes inverse cube root of each vector element.

```
IppStatus ippsInvCbrt_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvCbrt_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvCbrt_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsInvCbrt_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsInvCbrt_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Pow

Raises each element of the first vector to the power of corresponding element of the second vector.

```
IppStatus ippsPow_32f_A11 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsPow_32f_A21 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsPow_32f_A24 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsPow_64f_A50 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
IppStatus ippsPow_64f_A53 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
```

Powx

Raises each element of a vector to the constant power.

```
IppStatus ippsPowx_32f_A11 (const Ipp32f* pSrc1, const Ipp32f ConstValue,
   Ipp32f* pDst, int len);
```

```
IppStatus ippsPowx_32f_A21 (const Ipp32f* pSrc1, const Ipp32f ConstValue,
   Ipp32f* pDst, int len);
IppStatus ippsPowx_32f_A24 (const Ipp32f* pSrc1, const Ipp32f ConstValue,
   Ipp32f* pDst, int len);
IppStatus ippsPowx_64f_A50 (const Ipp64f* pSrc1, const Ipp64f ConstValue,
   Ipp64f* pDst, int len);
IppStatus ippsPowx_64f_A53 (const Ipp64f* pSrc1, const Ipp64f ConstValue,
   Ipp64f* pDst, int len);
```

Exponential and Logarithmic Functions

Exp

Raises e to the power of each vector element.

```
IppStatus ippsExp_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsExp_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsExp_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsExp_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsExp_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Ln

Computes natural logarithm of each vector element.

```
IppStatus ippsLn_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLn_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLn_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLn_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsLn_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Log10

Computes common logarithm of each vector element.

```
IppStatus ippsLog10_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLog10_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLog10_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsLog10_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsLog10_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Trigonometric Functions

Cos

Computes cosine of each vector element.

```
IppStatus ippsCos_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
```

```
IppStatus ippsCos_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCos_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCos_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsCos_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Sin

Computes sine of each vector element.

```
IppStatus ippsSin_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSin_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSin_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSin_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsSin_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

SinCos

Computes sine and cosine of each vector element.

```
IppStatus ippsSinCos_32f_A11 (const Ipp32f* pSrc, const Ipp32f* pDst1,
   Ipp32f* pDst2, int len);
IppStatus ippsSinCos_32f_A21 (const Ipp32f* pSrc, const Ipp32f* pDst1,
   Ipp32f* pDst2, int len);
IppStatus ippsSinCos_32f_A24 (const Ipp32f* pSrc, const Ipp32f* pDst1,
   Ipp32f* pDst2, int len);
IppStatus ippsSinCos_64f_A50 (const Ipp64f* pSrc, const Ipp64f* pDst1,
   Ipp64f* pDst2, int len);
IppStatus ippsSinCos_64f_A53 (const Ipp64f* pSrc, const Ipp64f* pDst1,
   Ipp64f* pDst2, int len);
```

Tan

Computes tangent of each vector element.

```
IppStatus ippsTan_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTan_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTan_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTan_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsTan_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Acos

Computes inverse cosine of each vector element.

```
IppStatus ippsAcos_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcos_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcos_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcos_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAcos_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Asin

Computes inverse sine of each vector element.

```
IppStatus ippsAsin_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsin_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsin_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsin_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAsin_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Atan

Computes inverse tangent of each vector element.

```
IppStatus ippsAtan_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtan_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtan_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtan_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAtan_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Atan2

Computes four-quadrant inverse tangent of elements of two vectors.

```
IppStatus ippsAtan2_32f_A11 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsAtan2_32f_A21 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsAtan2_32f_A24 (const Ipp32f* pSrc1, const Ipp32f* pSrc2,
   Ipp32f* pDst, int len);
IppStatus ippsAtan2_64f_A50 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
IppStatus ippsAtan2_64f_A53 (const Ipp64f* pSrc1, const Ipp64f* pSrc2,
   Ipp64f* pDst, int len);
```

Hyperbolic Functions

Cosh

Computes hyperbolic cosine of each vector element.

```
IppStatus ippsCosh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCosh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCosh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsCosh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsCosh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Sinh

Computes hyperbolic sine of each vector element.

```
IppStatus ippsSinh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSinh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSinh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsSinh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsSinh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Tanh

Computes hyperbolic tangent of each vector element.

```
IppStatus ippsTanh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTanh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTanh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsTanh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsTanh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Acosh

Computes inverse (nonnegative) hyperbolic cosine of each vector element.

```
IppStatus ippsAcosh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcosh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcosh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAcosh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAcosh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Asinh

Computes inverse hyperbolic sine of each vector element.

```
IppStatus ippsAsinh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsinh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsinh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAsinh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAsinh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Atanh

Computes inverse hyperbolic tangent of each vector element.

```
IppStatus ippsAtanh_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtanh_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtanh_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsAtanh_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsAtanh_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Special Functions

Erf
Computes the error function value.

```
IppStatus ippsErf_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErf_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErf_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErf_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsErf_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Erfc
Computes the complementary error function value.

```
IppStatus ippsErfc_32f_A11 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErfc_32f_A21 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErfc_32f_A24 (const Ipp32f* pSrc, Ipp32f* pDst, int len);
IppStatus ippsErfc_64f_A50 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
IppStatus ippsErfc_64f_A53 (const Ipp64f* pSrc, Ipp64f* pDst, int len);
```

Data Compression Functions

VLC and Huffman Coding Functions

VLCEncodeInitAlloc
Allocates memory and initializes the VLC encoder structure.

```
ippsVLCEncodeInitAlloc_32s (const IppsVLCTable* pInputTable, int
   inputTableSize, IppsVLCEncodeSpec_32s** ppVLCSpec);
```

VLCEncodeFree
Frees memory allocated for the VLC encoder structure.

```
void ippsVLCEncodeFree_32s (IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCEncodeInit
Initializes the VLC encoder structure.

```
ippsVLCEncodeInit_32s (const IppsVLCTable_32s* pInputTable, int
   inputTableSize, IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCEncodeGetSize
Computes the size of the VLC encoder structure.

```
ippsVLCEncodeGetSize_32s (const IppsVLCTable_32s* pInputTable, int
   inputTableSize, Ipp32s* pSize);
```

VLCEncodeBlock
Performs VLC encoding of a block of data.

```
ippsVLCEncodeBlock_16s1u (const Ipp16s* pSrc, int srcLen, Ipp8u** ppDst,
   int* pDstBitsOffset, const IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCEncodeOne
Performs VLC encoding of a single element.

```
ippsVLCEncodeBlock_16s1u (Ipp16s src, Ipp8u** ppDst, int*
   pDstBitsOffset, const IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCCountBits
Computes a number of bits required to encode the source block.

```
ippsVLCCountBits_16s32s (const Ipp16s* pSrc, int srcLen, Ipp32s*
   pCountBits, const IppsVLCEncodeSpec_32s* pVLCSpec);
```

VLCDecodeInitAlloc
Allocates memory and initializes the VLC decoder structure.

```
ippsVLCDecodeInitAlloc_32s (const IppsVLCTable_32s* pInputTable,
   int inputTableSize, Ipp32s* pSubTablesSizes, int numSubTables,
   IppsVLCDecodeSpec_32s** ppVLCSpec);
```

VLCDecodeFree
Frees memory allocated for the VLC decoder structure.

```
void ippsVLCDecodeFree_32s (IppsVLCDecodeSpec_32s* pVLCSpec);
```

VLCDecodeInit
Initializes the VLC decoder structure.

```
ippsVLCDecodeInit_32s (const IppsVLCTable_32s* pInputTable, int
   inputTableSize, Ipp32s* pSubTablesSizes, int numSubTables,
   IppsVLCDecodeSpec_32s* pVLCSpec);
```

VLCDecodeGetSize
Computes the size of the VLC decoder structure.

```
ippsVLCDecodeGetSize_32s (const IppsVLCTable_32s* pInputTable, int
   inputTableSize, Ipp32s* pSubTablesSizes, int numSubTables, Ipp32s*
   pSize);
```

VLCDecodeBlock
Decodes a block of VLC encoded data.

```
ippsVLCDecodeBlock_1u16s (const Ipp8u** ppSrc, int* pSrcBitsOffset,
   Ipp16s* pDst, int dstLen, const IppsVLCDecodeSpec_32s* pVLCSpec);
```

VLCDecodeOne
Decodes a single VLC encoded element.

```
ippsVLCDecodeOne_1u16s (const Ipp8u** ppSrc, int* pSrcBitsOffset,
   Ipp16s* pDst, const IppsVLCDecodeSpec_32s* pVLCSpec);
```

EncodeHuffInitAlloc

Allocates memory and initializes the structure for Huffman encoding.

```
IppStatus ippsEncodeHuffInitAlloc_8u(const int freqTable[256],
   IppHuffState_8u** ppHuffState);
```

HuffFree

Frees memory allocated for the Huffman encoding and decoding structures.

```
void ippsHuffFree_8u(IppHuffState_8u* pHuffState);
```

EncodeHuffInit

Initializes the Huffman encoding structure.

```
IppStatus ippsEncodeHuffInit_8u (const int freqTable[256],
   IppHuffState_8u* pHuffState);
```

HuffGetSize

Computes size of the external buffer for the Huffman encoding structure.

EncodeHuffOne

Performs Huffman encoding of a single element.

```
IppStatus ippsEncodeHuffOne_8u(Ipp8u src, Ipp8u* pDst, int
   dstOffsetBits, IppHuffState_8u* pHuffState);
```

EncodeHuff

Performs Huffman encoding.

```
IppStatus ippsEncodeHuff_8u(const Ipp8u* pSrc, int srcLen, Ipp8u* pDst,
   int* pDstLen, IppHuffState_8u* pHuffState);
```

EncodeHuffFinal

Performs Huffman encoding of the remainder.

```
IppStatus ippsEncodeHuffFinal_8u(Ipp8u* pDst, int* pDstLen,
   IppHuffState_8u* pHuffState);
```

HuffGetLenCodeTable

Returns the table containing lengths of Huffman codes.

```
IppStatus ippsHuffGetLenCodeTable_8u(int codeLenTable[256],
   IppHuffState_8u* pHuffState);
```

DecodeHuffInitAlloc

Allocates memory and initializes the Huffman decoding structure.

```
IppStatus ippsDecodeHuffInitAlloc_8u(const int codeLenTable[256],
   IppHuffState_8u** ppHuffState);
```

DecodeHuffInit

Initialized the Huffman decoding structure.

```
IppStatus ippsDecodeHuffInit_8u(const int codeLenTable[256],
   IppHuffState_8u* pHuffState);
```

DecodeHuffOne
Decodes a single code word.

```
IppStatus ippsDecodeHuffOne_8u(const Ipp8u* pSrc, int srcOffsetBits,
   Ipp8u* pDst, IppHuffState_8u* pHuffState);
```

DecodeHuff
Performs Huffman decoding.

```
IppStatus ippsDecodeHuff_8u(const Ipp8u* pSrc, int srcLen, Ipp8u* pDst,
   int* pDstLen, IppHuffState_8u* pHuffState);
```

HuffGetDstBuffSize
Computes the size of a destination buffer for Huffman encoding/decoding by blocks.

```
IppStatus ippsHuffGetDstBuffSize_8u(const int codeLenTable[256], int
   srcLen, int* pEncDstBuffSize, int* pDecDstBuffSize);
```

HuffLenCodeTablePack
Packs the table containing lengths of Huffman codes.

```
IppStatus ippsHuffLenCodeTablePack_8u(const int codeLenTable[256],
   Ipp8u* pDst, int* pDstLen);
```

HuffLenCodeTableUnpack
Unpacks the table containing lengths of Huffman codes.

```
IppStatus ippsHuffLenCodeTableUnpack_8u(const Ipp8u* pSrc, int* pSrcLen,
   int codeLenTable[256]);
```

Dictionary-Based Compression Functions

EncodeLZSSInitAlloc
Allocates memory and initializes the LZSS encoder state structure.

```
IppStatus ippsEncodeLZSSInitAlloc_8u (IppLZSSState_8u** ppLZSSState);
```

LZSSFree
Frees memory allocated for the LZSS state structure.

```
void ippsLZSSFree_8u (IppLZSSState_8u* pLZSSState);
```

EncodeLZSSInit
Initializes the LZSS encoder state structure.

```
IppStatus ippsEncodeLZSSInit_8u (IppLZSSState_8u* pLZSSState);
```

LZSSGetSize
Computes the size of the LZSS state structure.

```
IppStatus ippsLZSSGetSize_8u (int* pLZSSStateSize);
```

EncodeLZSS
Performs LZSS encoding.

```
IppStatus ippsEncodeLZSS_8u (Ipp8u**  ppSrc, int*  pSrcLen, Ipp8u**  p
   pDst, int* pDstLen, IppLZSSState_8u* pLZSSState);
```

EncodeLZSSFlush
Encodes the last few bits in the bitstream and aligns the output data on the byte boundary.

```
IppStatus ippsEncodeLZSSFlush_8u (Ipp8u** ppDst, int* pDstLen,
   IppLZSSState_8u* pLZSSState);
```

DecodeLZSSInitAlloc
Allocates memory and initializes the LZSS decoder state structure.

```
IppStatus ippsDecodeLZSSInitAlloc_8u (IppLZSSState_8u** ppLZSSState)
```

DecodeLZSSInit
Initializes the LZSS decoder state structure.

```
IppStatus ippsDecodeLZSSInit_8u (IppLZSSState_8u* pLZSSState);
```

DecodeLZSS
Performs LZSS decoding.

```
IppStatus ippsDecodeLZSS_8u(Ipp8u* ppSrc, int* pSrcLen, Ipp8u*8 ppDst,
   int* pDstLen, IppLZSSState_8u* pLZSSState);
```

EncodeLZ77Init
Initializes the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77Init_8u (IppLZ77comprLevel comprLevel,
   ippLZ77chcksm checksum, IppLZ77_8u* pLZ77State);
```

EncodeLZ77GetSize
Computes the size of the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77GetSize_8u (int* pLZ77VLCStateSize);
```

EncodeLZ77InitAlloc
Allocates memory and initializes the encoding state structure.

```
IppStatus ippsEncodeLZ77InitAlloc_8u (IppLZ77comprLevel comprLevel,
   IppLZ77chcksm checksum, IppLZ77_8u** ppLZ77State);
```

LZ77Free
Frees memory allocated for the LZ77 encoding and decoding structures.

```
void ippsLZ77Free_8u (IppLZ77State_8u* pLZ77State);
```

EncodeLZ77
Performs LZ77 encoding.

```
IppStatus ippsEncodeLZ77_8u (Ipp8u** ppSrc, int* pSrcLen, IppLZ77Pair**
   ppDst, int* pDstLen, IppLZ77Flush flush, IppLZ77State_8u*
   pLZ77State);
```

EncodeLZ77SelectHuffMode
Determines the optimal encoding mode.

```
IppStatus ippsEncodeLZ77SelectHuffMode_8u(IppLZ77Pair* pSrc, int srcLen,
   IppLZ77HuffMode* pHuffMode, IppLZ77State_8u* pLZ77State);
```

EncodeLZ77FixedHuff

Performs fixed Huffman encoding.

```
IppStatus ippsEncodeLZ77FixedHuff_8u (IppLZ77Pair** ppSrc, int* pSrcLen,
   Ipp8u** ppDst, int* pDstLen, IppLZ77flush flush, IppLZ77_8u*
   pLZ77State);
```

EncodeLZ77DynamicHuff

Performs dynamic Huffman encoding.

```
IppStatus ippsEncodeLZ77DynamicHuff_8u(IppLZ77Pair** ppSrc, int*
   pSrcLen, Ipp8u** ppDst, int* pDstLen, IppLZ77Flush flush,
   IppLZ77State_8u* pLZ77State);
```

EncodeLZ77StoredBlock

Stores the data block without compression.

```
IppStatus ippsEncodeLZ77StoredBlock_8u(Ipp8u** ppSrc, int* pSrcLen,
   Ipp8u** ppDst, int* pDstLen, IppLZ77Flush flush, IppLZ77State_8u*
   pLZ77State);
```

EncodeLZ77Flush

Writes the checksum and total length of the input data to the end of the stream.

```
IppStatus ippsEncodeLZ77Flush_8u (Ipp8u** ppDst, int* pDstLen,
   IppLZ77State_8u* pLZ77State);
```

EncodeLZ77GetPairs

Retrieves *pair data from the LZ77 encoding state structure*.

```
IppStatus ippsEncodeLZ77GetPairs_8u (IppLZ77Pair** ppPairs, int*
   pPairsInd, int* pPairsLen, IppLZ77State_8u* pLZ77State);
```

EncodeLZ77SetPairs

Sets pair data in the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77SetPairs_8u (IppLZ77Pair* pPairs, int pairsInd,
   int pairsLen, IppLZ77State_8u* pLZ77State);
```

EncodeLZ77GetStatus

Reads the deflate status value from the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77GetStatus_8u (IppLZ77VLCDeflateStatus*
   pDeflateStatus, IppLZ77_8u* pLZ77State);
```

EncodeLZ77SetStatus

Sets the deflate status to the desired value in the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77SetStatus_8u (IppLZ77DeflateStatus
   deflateStatus, IppLZ77_8u* pLZ77Stat);
```

EncodeLZ77Reset

Resets the LZ77 encoding state structure.

```
IppStatus ippsEncodeLZ77Reset_8u (IppLZ77State_8u* pLZ77State);
```

DecodeLZ77Init

Initializes the LZ77 decoding structure.

```
IppStatus ippsDecodeLZ77Init_8u(IppLZ77Chcksm checksum, IppLZ77State_8u*
   pLZ77State);
```

DecodeLZ77GetSize

Computes the size of the LZ77 decoding structure.

```
IppStatus ippsDecodeLZ77GetSize_8u(int* pLZ77StateSize);
```

DecodeLZ77InitAlloc

Allocates memory and initializes the LZ77 decoding structure.

```
IppStatus ippsDecodeLZ77InitAlloc_8u(IppLZ77Chcksm checksum,
   IppLZ77State_8u** ppLZ77State);
```

DecodeLZ77

Performs LZ77 decoding.

```
IppStatus ippsDecodeLZ77_8u(IppLZ77Pair** ppSrc, int* pSrcLen, Ipp8u**
   ppDst, int* pDstLen, IppLZ77Flush flush, IppLZ77State_8u*
   pLZ77State);
```

DecodeLZ77GetBlockType

Determines the type of encoded data block.

```
IppStatus ippsDecodeLZ77GetBlockType_8u(Ipp8u** ppSrc, int* pSrcLen,
   IppLZ77HuffMode* pHuffMode, IppLZ77State_8u* pLZ77State);
```

DecodeLZ77FixedHuff

Performs fixed Huffman decoding.

```
 IppStatus ippsDecodeLZ77FixedHuff_8u(Ipp8u** ppSrc, int* pSrcLen,
   IppLZ77Pair** ppDst, int* pDstLen, IppLZ77Flush flush,
   IppLZ77State_8u* pLZ77State);
```

DecodeLZ77DynamicHuff

Performs dynamic Huffman decoding.

```
IppStatus ippsDecodeLZ77DynamicHuff_8u(Ipp8u** ppSrc, int* pSrcLen,
   IppLZ77Pair** ppDst, int* pDstLen, IppLZ77Flush flush,
   IppLZ77State_8u* pLZ77State);
```

DecodeLZ77StoredBlock

Performs decoding of the block stored without compression.

```
IppStatus ippsDecodeLZ77StoredBlock_8u(Ipp8u** ppSrc, int* pSrcLen,
   Ipp8u** ppDst, int* pDstLen, IppLZ77State_8u* pLZ77State);
```

DecodeLZ77GetPairs

Retrieves *pair data from the LZ77 decoding state structure*.

```
IppStatus ippsDecodeLZ77GetPairs_8u(IppLZ77Pair** ppPairs, int*
   pPairsInd, int* pPairsLen, IppLZ77State_8u* pLZ77State);
```

DecodeLZ77SetPairs
Sets pair data in the LZ77 decoding state structure.

```
IppStatus ippsDecodeLZ77SetPairs_8u(IppLZ77Pair* pPairs, int pairsInd,
   int pairsLen, IppLZ77State_8u* pLZ77State);
```

DecodeLZ77GetStatus
Reads the inflate status value from the LZ77 decoding state structure.

```
IppStatus ippsDecodeLZ77GetStatus_8u(IppLZ77InflateStatus*
   pInflateStatus, IppLZ77State_8u* pLZ77State);
```

DecodeLZ77SetStatus
Sets the inflate status to the desired value in the LZ77 decoding state structure.

```
IppStatus ippsDecodeLZ77SetStatus_8u(IppLZ77InflateStatus inflateStatus,
   IppLZ77State_8u* pLZ77State);
```

DecodeLZ77Reset
Resets the LZ77 decoding state structure.

```
IppStatus ippsDecodeLZ77Reset_8u(IppLZ77State_8u* pLZ77State);
```

Adler32
Computes the Adler32 checksum for the source data buffer.

```
IppStatus ippsAdler32_8u (const Ipp8u* pSrc, int srcLen, Ipp32u*
   pAdler32);
```

CRC32
Computes the CRC32 checksum for the source data buffer.

```
IppStatus ippsCRC32_8u (const Ipp8u* pSrc, int srcLen, Ipp32u* pCRC32);
```

BWT-Based Compression Functions

BWTFwdGetSize
Computes the size of the external buffer for the forward BWT transform.

```
IppStatus ippsBWTFwdGetSize_8u(int wndSize, int* pBWTFwdBuffSize);
```

BWTFwd
Performs the forward BWT transform.

```
IppStatus ippsBWTFwd_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len,
   int* pIndex, Ipp8u* pBWTFwdBuff);
```

BWTInvGetSize
Computes the size of the external buffer for the inverse BWT transform.

```
IppStatus ippsBWTInvGetSize_8u(int wndSize, int* pBWTInvBuffSize);
```

BWTInv
Performs the inverse BWT transform.

```
IppStatus ippsBWTInv_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len, int
   index, Ipp8u* pBWTInvBuff);
```

BWTGetSize_SmalBlock

Computes the size of the external buffer for the BWT transforms for small data block.

```
IppStatus ippsBWTGetSize_SmallBlock_8u(int wndSize, int* pBuffSize);
```

BWTFwd_SmallBlock

Performs the forward BWT transform for small data block.

```
IppStatus ippsBWTFwd_SmallBlock_8u(const Ipp8u* pSrc, Ipp8u* pDst, int
   len, int* pIndex, Ipp8u* pBWTBuff);
```

BWTInv_SmallBlock

Performs the inverse BWT transform for small data block.

```
IppStatus ippsBWTInv_SmallBlock_8u(const Ipp8u* pSrc, Ipp8u* pDst, int
   len, int index, Ipp8u* pBWTBuff);
```

EncodeGITInitAlloc

Allocates memory and initializes the GIT encoding state structure.

```
IppStatus ippsEncodeGITInitAlloc_8u (int maxSrcLen, int maxDstLen,
   IppGITState_8u** ppGITState);
```

GITFree

Frees memory allocated for the GIT coding structures.

```
void ippsGITFree_8u (IppGITState_8u* pGITState);
```

EncodeGITInit

Initializes the GIT encoding state structure.

```
IppStatus ippsEncodeGITInit_8u (int maxSrcLen, int maxDstLen,
   IppGITState_8u* pGITState);
```

EncodeGITGetSize

Computes the size of the GIT encoding state structure.

```
IppStatus ippsEncodeGITGetSize_8u (int maxSrcLen, int maxDstLen, int*
   pGITStateSize);
```

EncodeGIT

Performs GIT encoding.

```
IppStatus ippsEncodeGIT_8u(const Ipp8u* pSrc, int srcLen, Ipp8u* pDst,
   int* pDstLen, IppGITStrategyHint strategyHint, IppGITState_8u*
   pGITState);
```

DecodeGITInitAlloc

Allocates memory and initializes the GIT decoding state structure.

```
IppStatus ippsDecodeGITInitAlloc_8u (int maxSrcLen, int maxDstLen,
   IppGITState_8u** ppGITState);
```

DecodeGITInit

Initializes the GIT decoding state structure.

```
IppStatus ippsDecodeGITInit_8u (int maxDstLen, IppGITState_8u*
   pGITState);
```

DecodeGITGetSize
Computes the size of the GIT decoding state structure.

```
IppStatus ippsDecodeGITGetSize_8u (int maxSrcLen, int* pGITStateSize);
```

DecodeGIT
Performs GIT decoding.

```
IppStatus ippsDecodeGIT_8u (const Ipp8u* pSrc, int srcLen, Ipp8u* pDst,
   int* pDstLen, IppGITStrategyHint strategyHint, IppGITState_8u*
   pGITState);
```

MTFInitAlloc
Allocates memory and initializes the MTF structure.

```
IppStatus ippsMTFInitAlloc_8u(IppMTFState_8u** ppMTFState);
```

MTFFree
Frees memory allocated for the MTF structure.

```
void ippsMTFFree_8u(IppMTFState_8u* pMTFState);
```

MTFInit
Initializes the MTF structure.

```
IppStatus ippsMTFInit_8u(IppMTFState_8u* pMTFState);
```

MTFGetSize
Computes the size of the MTF structure.

```
IppStatus ippsMTFGetSize_8u(int* pMTFStateSize);
```

MTFFwd
Performs the forward MTF transform.

```
IppStatus ippsMTFFwd_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len,
   IppMTFState_8u* pMTFState);
```

MTFInv
Performs the inverse MTF transform.

```
IppStatus ippsMTFInv_8u(const Ipp8u* pSrc, Ipp8u* pDst, int len,
   IppMTFState_8u* pMTFState);
```

EncodeRLE
Performs RLE encoding.

```
IppStatus ippsEncodeRLE_8u(const Ipp8u** ppSrc, int* pSrcLen, Ipp8u*
   pDst, int* pDstLen);
```

DecodeRLE
Performs RLE decoding.

```
IppStatus ippsDecodeRLE_8u(const Ipp8u** ppSrc, int* pSrcLen, Ipp8u*
   pDst, int* pDstLen);
```